TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

TURO VÄLIKANGAS

# SIMULATION METHOD DEVELOPMENT FOR FIN-AND-TUBE HEAT EXCHANGER WITH OPEN-SOURCE SOFTWARE

Master of Science thesis

Examiner: Reijo Karvinen
Examiner and topic approved by the
Faculty Council of the Faculty of
Engineering Sciences
on 12th August 2015

# ABSTRACT

**TURO VÄLIKANGAS**: Simulation method development for Fin-and-Tube Heat Exchanger with Open-source software
Tampere University of Technology
Master of Science thesis, 73 pages, 15 Appendix pages
August 2015
Master's Degree Programme in Mechanical Engineering
Major: Flow Dynamics
Examiner: Prof. Reijo Karvinen
Keywords: heat transfer, simulation, fin-and-tube, x-slit, open source, Salome, OpenFOAM, Paraview, snappyHexMesh

This thesis project was done in collaboration with Chalmers University of Technology under the supervision of Professor Håkan Nilsson.

In this thesis project, a numerical simulation method for fin-and-tube heat exchanger is developed. All the programs used for any parts of the simulation process are distributed as open source software. This study focuses on the air side convective heat transfer simulations. The simulation phases that were developed are the geometry generation, meshing, solving the fields and post-processing of the computed data. Two different fin shapes were studied. The first fin shape was one with available experimental values to compare with and then another fin shape was selected for its high meshing difficulty. It was found that even though the plain fin has less vorticity in the flow field it changes heat with a better efficiency parameter compared to the x-slit fin because of its smaller tube diameter and more compound design.

# TIIVISTELMÄ

**TURO VÄLIKANGAS**: Putkilamellilämmönsiirtimen simulointimenetelmän kehitys avoimen lähdekoodin ohjelmistoilla
Tampereen teknillinen yliopisto
Diplomityö, 73 sivua, 15 liitesivua
Elokuu 2015
Konetekniikan koulutusohjelma
Pääaine: Virtaustekniikka
Tarkastaja: Prof. Reijo Karvinen
Avainsanat: lämmönsiirto, simulointi, putkilamelli, x-slit, avoin lähdekoodi, Salome, OpenFOAM, Paraview, snappyHexMesh

Tämä diplomityö on tehty yhteistyössä Chalmerssin teknillisen yliopiston kanssa, jossa tarkastajana toimi Professori Håkan Nilsson.

Tässä diplomityöprojektissa kehitettiin numeerinen simulointimenetelmä, jolla on mahdollista tutkia putkilamellilämmönsiirtimien lämmönsiirto- ja painehäviöominaisuuksia. Kaikki ohjelmat, joita simulointiprosessissa on käytetty ovat avoimen lähdekoodin ohjelmistoja. Pääasiassa tässä diplomityössä keskitytään tutkimaan putkilamellilämmönsiirtimen ilmapuolen konvektiivista lämmönsiirtoa. Simulointimenetelmässä kehitetyt eri työvaiheet ovat geometrian luonti, verkotus, virtauskenttien laskenta ja syntyneen tiedon jälkikäsittely. Tässä diplomityössä tutkittiin kahta erilaista lamelligeometriaa. Ensimmäinen niistä oli suora lamelligeometria, johon oli saatavilla mittausdataa, jolla mallin validionti voitiin suorittaa. Toinen mallinnettu lammeligeometria on nimeltään x-slit fin, joka valikoitui toiseksi lamellivaihtoehdoksi koska sen ympärille muodostuvan virtausalueen verkottaminen oli oletettavasti erittäin vaikeaa.

Diplomityön tuloksena voidaan todeta, että putkilamellilämmönsiirtimen mallintaminen on mahdollista avoimen lähdekoodin ohjelmistoilla. Simulointien tuloksena todettiin, että vaikka suora lamelligeometria luo vähemmän pyörteitä virtauskenttään, se siirtää lämpöenergiaa paremmalla hyötysuhdeparametrilla verrattuna x-slit geometriaan, koska sen putkien koko oli pienempi ja putkien sijoittelu oli lähempänä toisiaan.

# PREFACE

First of all I want to thank Professor Reijo Karvinen for originally providing this project for me and trusting me to do the work in Chalmers Technical University. Second I want to thank Professor Håkan Nilsson for being my supervisor, introducing me to the world of OpenFOAM and introducing me to the department of Fluid dynamics at Chalmers. Then I want to thank M.Sc. Taru Lähteenmäki for choosing me to do this project and all the guidance I have got during the thesis.

The amount of progress I made in the development of this simulation model would not have been possible without the help and guidance of Antti Mikkonen, Mikko Folkersma and Ardalan Javadi. Especially the help from Antti and Mikko was a crucial part of the success of the whole project. I also want to thank all the other people at Chalmers Technical University for the trips, company and brainstorming sessions that I was taken part in during my time in Gothenburg.

I want to also thank my little sisters, parents and grandparents for the financial and emotional care during the years I have spent in the University, especially thank you for taking care of my stuff during the times I have being on an exchange. I want to thank all the old and new friends and especially Salami group: Janne Kivinen, Jesse Niemi, Matias Salonen, Pauli Siivonen and Riku Lehto for the peer support during these years and LINNUT group for taking care of that I do not forget where I come from. And finally I want to thank my girlfriend Anna Kaipanen for listening, being supportive and taking my mind off from CFD-related topics when needed.

Tampere, 25.8.2015

Turo Välikangas

# TABLE OF CONTENTS

# 1. INTRODUCTION

In this Master's thesis a numerical simulation method for fin-and-tube heat exchangers is developed. Simulations are done in computational means with using only open source software. Fin-and-tube heat exchangers are commonly used heat exchanger type in air conditioning mostly because of its flexible manufacturing process. The capacity and characteristics of the heat exchanger can be scaled up or down according to the requirements of the application. Fin-and-tube heat exchanger is constructed from fin plates that are penetrated with tubes that are connected to each other with a header from start and finish of the circuit. Heat is transferred between the fluid that circulates inside the tubes and the air that flows between the fins. In figure 1.1 is an illustration of different types of fin-and-tube heat exchangers.

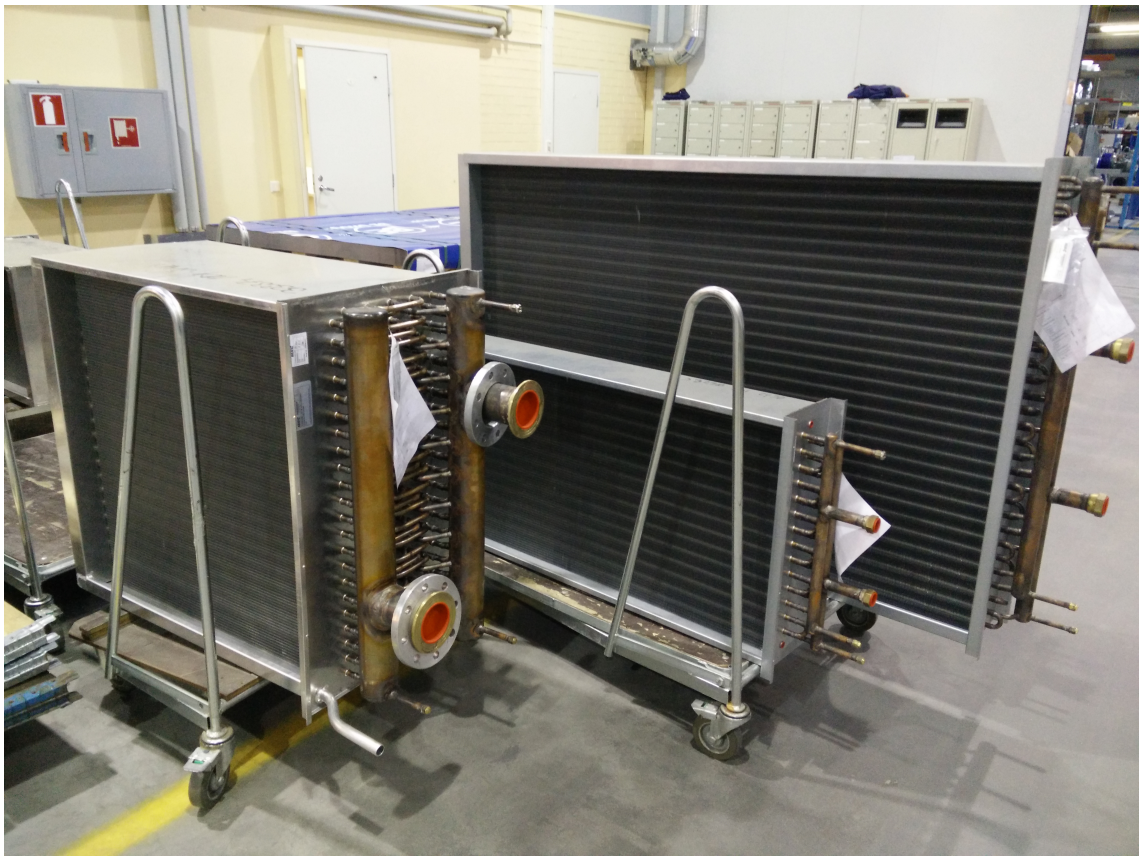


***Figure 1.1*** *Different sizes of fin-and-tube heat exchangers are designed for different applications*

Different sizes of heat exchangers are used in different situations. The more face front area the heat exchanger has the more air can go through the heat exchanger in a time period. On the other hand, more depth in the flow direction leads into higher possible change in the temperature of the air. The amount and length of the circuits defines also how much power is required or how efficiently the heat would like to be transferred. All these and a lot more smaller design parameters define the characteristics of the heat exchanger.

For the simulations in this thesis an open source simulation program called OpenFOAM (Open Field Operation And Manipulation) is used to calculate the amount of heat that is transferred to the air and how much of its pressure is lost when it flows through the flow medium. OpenFOAM uses finite-volume method to represent partial differential equations in the form of algebraic equations and solves them with respect to the boundary conditions. In the creation of all the flow mediums a python script based geometry modelling was used with Salome open source

CAD software. For meshing, an open source meshing tool called snappyHexMesh and a meshing module in Salome was used. For post-processing the data calculated with OpenFOAM, again another open source program called Paraview was used. All these programs are distributed under a GNU General public license which means that "Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed" (gpl, 2015). This means that making Computational Fluid Dynamics(CFD) simulations with these programs is free. Anyone is also free to change and distribute the software with financial gains but the source code must also be supplied.

## 1.1 Fin-and-tube heat exchanger

A good heat exchanger transfers heat between two objects with the lowest possible cost, in respect to the laws of nature. For different heat exchangers the cost of using them can occur in different forms. For fin-and-tube, the cost can be seen in the pressure drop of the fluid flowing inside the tubes and also in the pressure drop of the air flowing through the tube banks. This pressure needs to be created with a fan or a pump to facilitate the fluid flow, the cost of this is energy. Fin-and-tube heat exchanger's efficiency can be measured as the ratio between how much energy it transfers respect to the pressure drop that is encountered by the air.

### 1.1.1 Literature Survey

Before the 21-century most of the studies done on the research of fin-and-tube heat exchangers were done with experimental means (for example Goldstein and Sparrow (1976), Jang et al. (1996), Wang et al. (1998) and Wang et al. (1999b) ) and the biggest problem with this choice of research approach was with the repeatability of the experimental set up. Bad documentation of some small but crucial details on the wind tunnel or other unintentional differences in the inlet conditions leads to a point where different experiments with same geometries led to difference in the results (Webb, 2005). This is why the comparison of the effect of a change in one parameter to the final results was impossible to proof explicitly. This of course forced researchers into making correlations of a collection of experiments (for example Wang et al. (2000a), Wang et al. (1999a) and Wang et al. (2002a) ) done on a same heat exchanger geometry and then these correlations are used in a Air Handling Unit(AHU) selection program in the everyday work of design engineers.

Then when CFD-methods became more and more popular, and more importantly, bigger computational resources where in the reach of most universities, the computational research of fin-and-tube heat exchangers started and has continued until these days. With CFD-means all the design parameters for example fin spacing (Romero-Méndez et al., 2000) (Mon and Gross, 2004), fin thickness (Lu et al., 2011) or the location and size of the tubes (Erek et al. (2005) Tutar and Akkoca (2004) and Ryu et al. (2014)) can be studied separately keeping all the other variables constant. Therefore the causal link between changes was clear and measurable.

### 1.1.2 Enhancement methods

When it comes to the fin of the fin-and-tube heat exchanger, a lot of studies has been done both experimentally and numerically for many different fin shapes to enhance the heat transfer. Most traditional ones are the herringbone (Wang et al., 2002a) and sine-wave (Choi et al., 2013) fin shapes which main enhancement method was to create transverse vortices to increase the heat transfer. This of course increases the pressure drop because of the recirculation zone in the flow field. Then some perforated fin geometries for example louvered Wang et al. (1998) and slits (Wang et al., 1999c), which are usually referred to as compact fin shapes (Wang et al., 2001) are used in the aim to save in the material costs of making the heat exchangers. Most contemporary methods on the fin side heat transfer enhancement are the Vortex Generators(VG) (Leu et al. (2004), Wang et al. (2002b) and Joardar and Jacobi (2008)) that are used to create longitudal vortexes. This is beneficial compared to other enhancement methods because the created vortices are longitudinal and therefore the increased pressure drop is lower for the same amount of enhanced heat transfer.

Other main heat transfer enhancement method is to modify the inside of the tubes. The idea is to increase the vorticity of the fluid flowing inside the tubes in a way that it creates as small pressure drop as possible. These are called inner grooving of the tubes(Tang et al., 2009), Discrete Double Inclined Ribs (Li et al., 2007), v-shaped micro-groove (Chiang and Esformes, 1994) and other shapes that can be created in the mechanical expansion process.

Finally it is important to think about other matters outside the heat exchanger fin design for example the header, circulation design and the placement of the fan, filters and other appliances in the AHU in pursuit for the best efficiency of the heat exchanger.

Despite of all the research that has been done both experimentally and numerically, no generic best possible option or combination has been found. This is because when a design variable is optimized for the specific set up of the heat exchanger it is also a function of other variables and when these change, the previous "best" option is no longer the best one. The object of the optimization can also change from an application to the other. Also environmental variables like condensation and freezing of the heat exchanger brings another set of questions to the table that might change the weight of different variables in the design process.

### 1.1.3 Measurements Vs Computational Fluid Dynamics (CFD)

The main problem of measuring heat transfer in a heat exchangers is the question of repeatability of the environmental variables in the testing procedure. These can be for example the humidity and temperature of the incoming air in the experiments. Making sure that these factors stay constant with different measuring set ups is extremely difficult without a professional wind tunnel environment. Heat transfer characteristics can deviate a lot for different flow situations and therefore when comparing different designs it is essential that the environmental variables stay the same.

A CFD approach offers a big improvement overall in the study of heat transfer due to its repeatability and reliability. With CFD means, many different design aspects can be compared in a short period of time if necessary computational resources are available. Commercial CFD programs can though cost a huge amount of money. The whole industry of CFD programs is a fairly new business and right now it seems that it is dominated by only few players. Few of the biggest software houses have bought all the small players on the market and integrated them in to their own programs. This has caused the price of the licences to increase so high that one license needs at least two engineers to fully exploit the fee and produce enough simulations to pay up for these costs. This is of course if the question of finding a customer and the price for these simulations is taken care of. One year licence with access to the help desk services can cost up to 60000$ and a full ownership of the license, including the same one year service package, can be as high as 80000$. These are of course obscure numbers if you take into account that the total cost of a full time engineer can reach this same level. Of course one engineer usually has to spend some time with other matters also such as reporting, presenting, measuring and etc. not to mention weekends or holidays. It is clear that the efficient use of one license

by a single engineer can never be achieved. This is why open source alternatives are coming more and more popular especially for small scale businesses. Open source alternatives such as OpenFOAM are also highly customisable which might already make it a clear choice for some companies that need a lot of simulations on a one specific subject.

# 2. THEORY

In this chapter some fundamental aspects of heat transfer, thermal resistance and especially the effect of the combination of flow and heat distribution on heat transfer is illustrated.

## 2.1 Fundamentals of Heat Transfer

Heat transfer can be divided into three different modes: conduction, convection and radiation, where only first two are considered in this paper due to the low temperature levels in fin-and-tube heat exchangers. The law that governs heat conduction was first proposed by J.B. Fourier in 1822 (Mills, 1999). It states that the heat will flow through the medium in the direction of decreasing temperature gradient. Therefore the one dimensional heat flux can be defined as:

$$\dot{q} = -k\frac{dT}{dx} \tag{2.1}$$

where k is called the thermal conductivity of the material. The minus sign comes from the definition that when the gradient is negative the heat flux is positive in the direction of positive x-axis (Mills, 1999).

Convective heat transfer is a combination of conduction and advection. Advection can be described as the measure of mass that is transferred with the flow field. Convection is then an integration of conduction and advection in the flow medium. The amount of heat that is transferred from a surface to the flowing medium can be described with a heat transfer coefficient that is defined as:

$$h = \frac{\dot{q}}{A\Delta T} \tag{2.2}$$

which is the amount of heat transferred $\dot{q}$ per the surface area A and the temperature difference $\Delta T$.

### 2.1.1 Thermal circuit analogy

When considering the different parts of the thermal circuit in a fin-and-tube heat exchanger, where heat is transferred between the fluid inside the tubes and the air flowing through, it is important to outline how big contribution does different thermal resistances have. Thermal resistance is the reverse of the thermal conductivity or heat transfer coefficient which means that a high heat transfer coefficient leads to low value of thermal resistance. This helps in the process of figuring out which parts should be paid the most attention in the attempt of increasing the efficiency of the whole process. In figure 2.1 the whole thermal circuit of a fin-and-tube heat exchanger is shown.

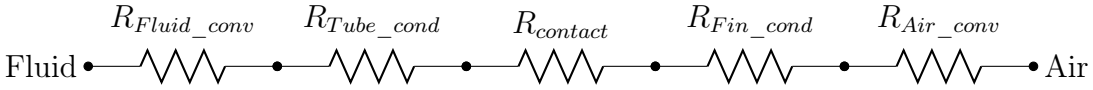



***Figure 2.1*** *Thermal circuit analogy in the heat transfer process in fin-and-tube heat exchanger*

In the figure 2.1, $R_{Fluid\_conv}$ is the thermal resistance of the convective heat transfer from the fluid inside of the tube to the tube itself, $R_{Tube\_cond}$ is the thermal resistance of the conduction heat transfer through the tube, $R_{contact}$ is the conduction and convection heat transfer resistance from the tubes to the fins, $R_{Fin\_cond}$ is the thermal resistance of the conduction in the fin and $R_{Air\_conv}$ is the convective heat transfer from the fins to the air. It has been studied that the air side convective thermal resistance $R_{Air\_conv}$ can be said to be around 55-65% from the overall resistance in the whole thermal resistance circuit in a fin-and-tube heat exchanger. The contact resistance $R_{contact}$ and thermal convective resistance on the fluid side $R_{Fluid\_conv}$ have proportionally equal contribution to the resistance of the thermal circuit with around 15-25% each. This leaves only few percent for the conduction in the tubes (Wang et al. (2000b) and Jeong et al. (2006)). These values indicate that most potential efficiency increase can be achieved by enhancing the heat transfer on the convective air side.

### 2.1.2 Boundary layer

The velocity and temperature profiles on the wall are the most important factors in heat transfer. Convective heat transfer can be said to be the integration of these

profiles over the flow medium. The result of this integration is larger if the slope of these profiles stays as normal to the wall as possible as close to the wall as possible. In other words the closer the dashed line, that marks the end of the boundary layer, is in figure 2.2, the higher the heat transfer is between the surface and the fluid flowing over it.

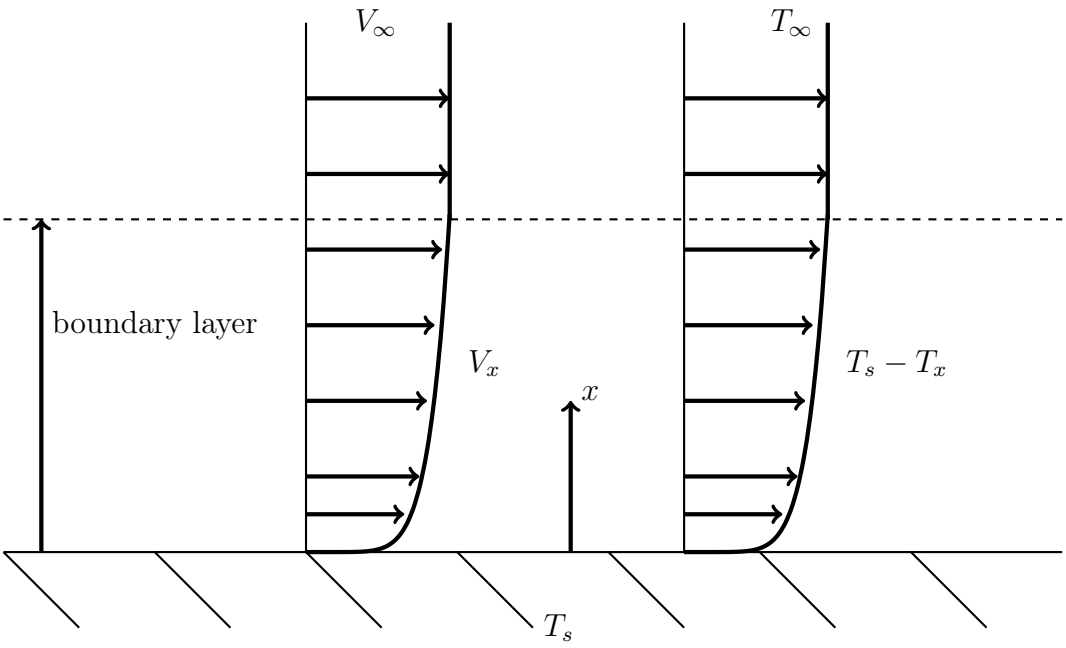



***Figure 2.2*** *Velocity and temperature profiles of a fluid near the wall*

The boundary layer is considered to be defined as the layer next to the wall where the speed and temperature is lower (or higher in the case for the temperature) than free stream values as illustrated in figure 2.2 with a dashed line. The shape of these profiles is a function of the free stream speed $V_\infty$ and temperature $T_\infty$ as well as the surface roughness. Another factor that influences the shape of these profiles and is almost independent from the variables mentioned earlier is the Prandtl number which is defined as:

$$Pr = \frac{\nu}{\alpha} \tag{2.3}$$

where $\nu$ is the viscous and $\alpha$ is the thermal diffusivity. For air the value of Prandtl number is around $Pr = 0.7$ for a wide temperature range.

## 2.2 Performance parameters

For better comparison of the overall performance of different fin shapes and a simpler way to present all the numerical 3d field data, it is convenient to define some non-dimensional parameters for the illustration of the results. First one that needs to be defined is the Reynolds number, which for the channel between the fins is defined according to Reynolds (1883) is as:

$$Re_{D_h} = \frac{u_{max} D_h}{\nu} \tag{2.4}$$

in which case $u_{max}$ is the velocity of the air in the minimum cross-sectional area, $\nu$ is the dynamic viscosity of the air and the hydraulic length scale $D_C$ according to Fornasieri and Mattarolo (1991) can be laid out as:

$$D_h = \frac{4(F_p - t)(P_t - Dc)P_l}{2(P_l P_t - \pi D_c^2/4) + \pi D_c(F_p - t)} \tag{2.5}$$

where $F_p$ is the fin pitch, $t$ is the thickness of the fin, $P_t$ is the transverse distance, $P_l$ is the longitudinal distance of the tubes and $D_c$ is the outside diameter of the collar around the tubes. An illustration of the geometry parameters can be seen later in section 4.1.1. Next we define the non-dimensional number called the Nusselt number as described by Mills (1999), which defines how many times does the advection of the flow field enhances heat transfer compared to the situation where the flow field would stand still and heat would only propagate through the air with conduction. Nusselt number used here is the global Nusselt number defined as:

$$Nu = \frac{h}{k_i/D_h} \tag{2.6}$$

where $h$ is the area averaged heat transfer coefficient and $k_i$ is the thermal conductivity of the flowing fluid medium. The heat transfer coefficient is calculated according to Mills (1999) as follows:

$$h = \frac{Q}{\eta_0 A_t \Delta T_{lm}} \tag{2.7}$$

where $Q = \dot{m} Cp(T_{in} - T_{out})$ is the overall transferred heat capacity, $\dot{m}$ is the mass flow, $C_p$ is the specific heat capacity, $T_{in}$ and $T_{out}$ are the mass averaged temperature

of the air in the inlet and outlet, respectively. Other members in 2.7 are the fin efficiency and total heat transfer area defined in subsection 2.2.1 and the last one is the logarithmic mean temperature. Logarithmic mean temperature is defined because of the uncertain distribution of the temperature difference between the air and the fin. It is clear that the temperature of the air changes as it flows through the channel and no constant temperature value can not be defined for the whole channel that would give the correct temperature difference and therefore the correct amount of heat transferred. This is why a logarithmic mean temperature difference is defined as follows:

$$\Delta T_{lm} = \frac{(T_{in} - T_w) - (T_{out} - Tw)}{ln[(T_{in} - T_w)/(T_{out} - T_w))]} \tag{2.8}$$

where the $T_{in}$, $T_{out}$ and $T_w$ are the temperatures of the air at the inlet and outlet and the temperature of the wall, respectively.

Then finally for the normalised non-dimensional heat transfer in a fin-and-tube heat exchanger a Colburn j-factor (Geankoplis, 2003) can be defined as:

$$j = \frac{h}{\rho u_{max} C_p} Pr^{2/3} \tag{2.9}$$

where all the other members are explained before.

Then for the normalised non-dimensional pressure drop in the flow channel a fanning friction factor according to Wang et al. (1996) is defined as:

$$f = \frac{\Delta p}{\frac{1}{2}\rho u_{max}^2} \times \frac{A_c}{A_t} \tag{2.10}$$

where the new variable $A_c$ is minimum cross-sectional area.

### 2.2.1 Fin efficiency

Because only the convection on the air side of the heat exchanger is studied in this study, the conduction heat transfer in the fin is left out of the simulations. This means that the whole fin is set to a constant temperature in the numerical model and no change in temperature distribution exists. In the fully realistic case, the temperature near the tubes would be close to the temperature of the fluid that

flows inside the tubes and the distribution of the temperature will change gradually towards to the temperature of the incoming air and reaches its peek in the point that is the farthest away from the tubes. This of course is extremely hard to estimate and because adding the coupled heat transfer simulation into the time frame of the thesis is impossible, this has to be estimated with another method. For this, a concept called fin efficiency is applied to the calculations to include the fact that in the real case, where the experiments are from, the fin is not always on the same temperature as the liquid flowing in the tubes. Fin efficiency describes how many percent the real distribution transfers heat compared to the constant temperature fin. Fin efficiency used in the fin-and-tube heat exchanger is defined as a area weighted fin efficiency:

$$\eta_0 = 1 - \frac{A_f}{A_t}(1 - \eta_f) \tag{2.11}$$

where $A_f$ is the fin surface area, $A_t$ is the total heat transfer area(fins and tubes) and $\eta_f$ is the fin efficiency according to Schmidt (1949) defined as:

$$\eta_f = 1 - \frac{tanh(mr_{eq}\psi)}{mr_{eq}\psi} \tag{2.12}$$

where $m = \sqrt{(2h/k_f t)}$, $\psi = (R_{eq}/r_{eq}-1)[1+0.35ln(R_{eq}/r_{eq})]$, $R_{eq} = 1.27X_M(X_L/X_M - 0.3)^{0.5})$, $r_{eq} = c/2\pi$, $X_L = 0.5[(P_t/2)^2 + P_l^2]^{0.5}$ and $X_M = P_t/2$, where all the members correspond to the variables explained in section 2.2 but $k_f$ is in this context the thermal conductivity of the fin material.

# 3. MODEL DESCRIPTION

In this chapter different theories, concepts and assumptions that are made in the process of modeling the air going through a fin-and-tube heat exchanger are illustrated and most important ones explained in detail. First the theory of continuum hypothesis is introduced and then the governing equations that are used to model the flow are opened up and important features are emphasized.

## 3.1 Continuum hypothesis

The concept of treating fluids as continuous material, where no clear distinction between fluid particles and molecules can not be made, is called *continuum hypothesis*. The flow characteristics that underline this phenomena are that the time and length scales are much smaller than what we as humans are used to feel and comprehend. If we consider for example the air in under atmospheric conditions we can say that the average spacing and collision time between the molecules are $3 \times 10^{-9}$ and $10^{-10}$, respectively. When comparing to the smallest length scales in real flow situations they are rarely under 0.1mm. In an extreme flow situation the flow speed can be up to several hundred meters per second. This would lead in to timescales up to $10^{-6}$ which is still several order of magnitudes bigger than the ones between the molecules. Therefore we can say that the continuum hypothesis is a valid assumption when analysing the motion of air in atmospheric conditions (Pope, 2000).

## 3.2 Modelling of the flow

The equation that describes the motion of incompressible newtonian flow was found by C.L.M.H Navier (1785-1836) and Sir George G.Stokes (1819-1903) independently from each other back in the 1800 (Frank M, 2003). These equations are discretized for a number of calculation points in the flow medium and then solved with numerical means. This can be said to be the fundamental idea behind CFD.

## 3.3 Governing equations

The governing equations for the air-flow inside the heat exchanger are steady state continuity, Navier-Stokes for momentum conservation, energy and scalar transport equations in three Cartesian directions.

For all the simulations in this thesis a rhoSimpleFoam-solver was chosen. This solver is a compressible solver with added transport equation for temperature, but the effect of buoyancy is left out of the equations to save in computational expenses. It is therefore important to look in to more detail of the solved equations that the solver resolves to understand the underlying physics that governs the flow.

### 3.3.1 Continuity Equation

The continuity equation ensures that the simulation process obeys the *law of conservation of mass* which means that no mass can be created or disappear in the flow medium.

$$\frac{\partial(\rho u_i)}{\partial x_i} = 0 \tag{3.1}$$

where $\rho[\frac{kg}{m^3}]$ denotes the density of the fluid particle and $u_i[\frac{m}{s}]$ denotes the velocity components in all three directions when summed over the dummy index $i$.

### 3.3.2 Momentum Equation

Momentum equation or commonly referred *Navier-Stokes* equation states that viscous stresses are proportional to the element strain rates and the viscous coefficient for newtonian fluids according to Frank M (2003).

$$\frac{\partial(\rho u_i u_j)}{\partial x_j} = \frac{\partial}{\partial x_i}(\mu \frac{\partial u_j}{\partial x_i}) - \frac{\partial p}{\partial x_i} \tag{3.2}$$

where $\rho$, $u_i$ and $u_j$ are the same as referred in eq. 3.1. Where as $\mu$ $[\frac{m^2}{s}]$ denotes the *dynamic (shear) viscosity* and $p$ is the static pressure $[\frac{N}{m^2}]$.

### 3.3.3 Energy Equation

The energy transportation equation for a steady state, non-reactive (no source term), compressible (the internal energy is only a function of specific heat $C_p$ and temperature T), calorically perfect (due to low temperatures the specific heat can be considered constant) gas with neglected dissipation. Air can be considered to be a gas without dissipation (kinetic energy transformed to heat) but for example oil in a car gear box, can not. For the air in a fin-and-tube heat exchanger we get

$$\frac{\partial}{\partial x_i}(u_i T) = \frac{\partial}{\partial x_i}(\alpha \frac{\partial u_j}{\partial x_j}) \tag{3.3}$$

where T and $\alpha$ are the temperature and the thermal diffusivity of the air, respectively. Thermal diffusivity $\alpha$ $[\frac{m^2}{s}]$ which is the measure of thermal inertia that describes how fast temperature concavities are smoothen out in medium can also be expressed as $\alpha = \frac{k}{\rho C_p}$ (Venkanna, 2010). Therefore the energy equation can be written in a similar way as momentum equation:

$$\frac{\partial}{\partial x_i}(\rho u_i T) = \frac{\partial}{\partial x_i}(\frac{k}{C_p} \frac{\partial u_j}{\partial x_j}) \tag{3.4}$$

### 3.3.4 Transport equation for an arbitrary scalar

The same way as the energy equation 3.4 describes the transport of thermal energy, other scalar attributes and their diversion in the flow field can be described in a similar manner and then a transport equation for a scalar $\phi$ is denoted as:

$$\frac{\partial}{\partial x_i}(\rho u_i \phi) = \frac{\partial}{\partial x_i}(\Gamma_\phi \frac{\partial u_j}{\partial x_j}) \tag{3.5}$$

where $\phi$ is the transported scalar in the flow field. $\Gamma_\phi$ and $S_\phi$ on the other hand are the diffusion coefficient and the source term for specific scalar, respectively.

# 4. CFD ANALYSIS

In this chapter all the phases that need to be completed in the simulation of the fin-and-tube heat exchanger with numerical calculation software are explained in detail. First the geometry of the flow medium is created, then it will be meshed with two different methods, then calculated and finally the values are illustrated and post-processed for further analysis.

## 4.1 Geometry modelling

In this thesis the first geometry was created by using an open source software called Salome. The geometries were created by hand by defining first points and lines in three dimensional space and then creating two dimensional faces and then finally three dimensional bodies from them. All the geometries were created with python scripts that are written with the library commands provided by the Salome python interface modules. An example of the python code for creating the model validation case is provided in the appendix A. This way the full potential of parametrized geometry creation can be achieved for the purpose of the optimisation of the heat exchangers in the future. Two different types of geometries were created. The first geometry created was the actual shape of the fluid domain that was then meshed as it is for the simulations and it does not include the air in front of the fin. It should be noted here that the entrance effect of the fin is therefore not included in the flow domain. The second geometry on the other hand is the shape of the fin itself which was then cut out from the background mesh to form the fluid medium as explained later in 4.1.2. To revise, in the first approach the flow medium itself was created but in the second approach the fin shape, that forms the boundary of the flow medium, was created. The decision of what part actually should be modelled when simulating the fin-and-tube heat exchanger is a difficult one. The question is how many rows of tubes should be modelled to achieve a simulation that includes all the flow characteristics in this type of flow situation. It has been shown with experiments by Wongwises and Chokeman (2005) as well as with computational means by Xie

et al. (2009) that the overall j- and friction factor decreases as the number of tube rows increases and that the value seems to saturate after 6 tube rows. In this thesis, computational resources restrict the coverage of different aspects studied, therefore the effect of the number of tube rows is left for future investigations. Another important cropping decision is to decide how much of the flow is simulated in width and height directions. One flow path between the fins and tubes is chosen to be modelled in this thesis, since it includes all the hydrothermal characteristics of the geometry. Finally the decision of where to draw the line vertically, where the flow domain starts and where it ends, comes to picture and the choices are:

1. Model the medium between two fin plates, option 1. in figure 4.1
2. Model half of the medium on both sides of the fin, option 2. in figure 4.1

The choice between these two is purely dependent on the simulation method used and how the mesh and boundary conditions are set for specific case.

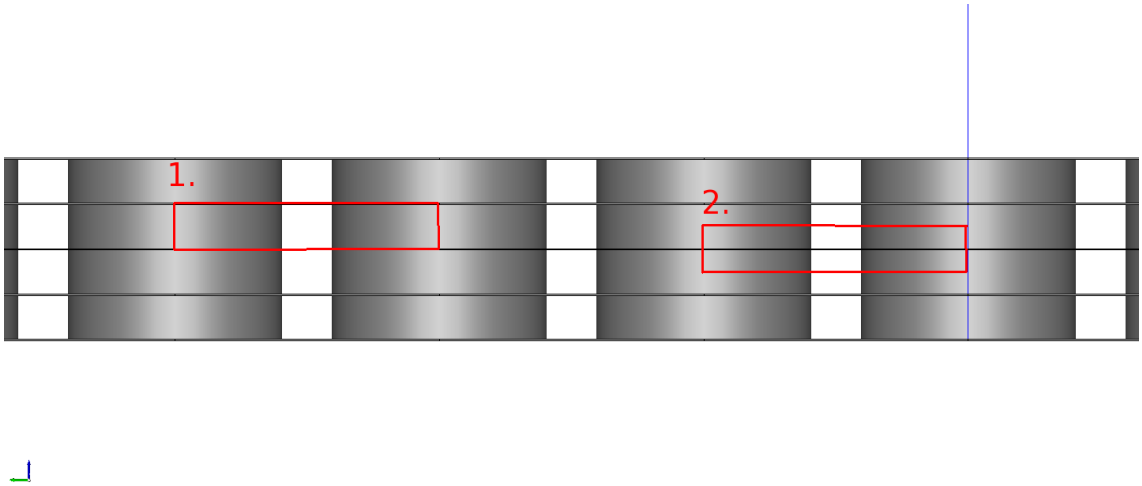



***Figure 4.1*** *Figure of two different options for modelling the heat exchanger*

This time the choice number 1. was chosen and so the air flowing between two fins is modelled with one half of the fin on both upper and lower side of the flow medium. This choice leads to few specific problems that were not seen beforehand, which will be explained later in this chapter.

### 4.1.1 Fin shape

In this thesis a simulation method is developed for two different fin shapes. First one illustrated in figure 4.2 is a plain straight fin that has no corrugations, slits, louvers or perforated holes. This was chosen as the Case 1, because of its simple design and the availability of experimental data provided by Wang et al. (1996).

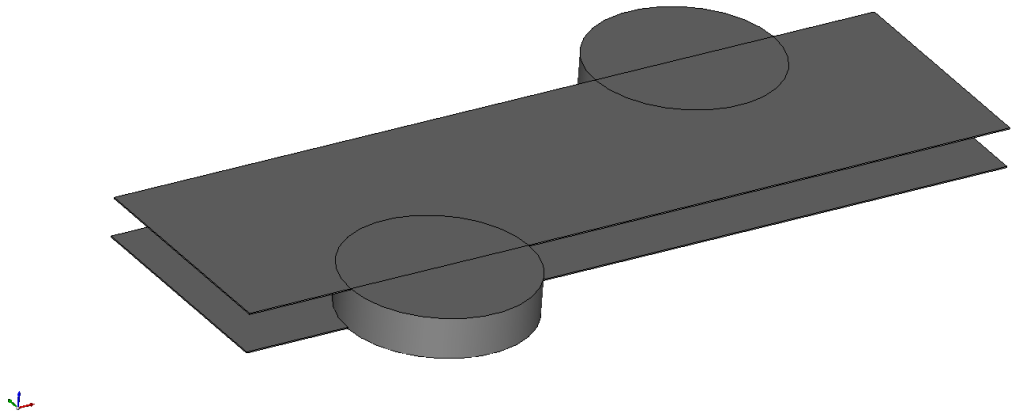


***Figure 4.2*** *The shape of the fin for the model validation case*

In a conventional heat transfer enhancement, for example the herringbone fin, the shape of the fin creates a vortex which recirculation direction is opposite to the main direction of the flow, the recirculation zone can then be said to be a latitudinal vortex. On the other hand Vortex Generators create mainly longitudinal vortexes where the circulation zones are in a plane which is perpendicular to the main direction of the flow. Therefore it creates lower pressure drop for the same heat transfer enhancement.

For now we leave the study of VG:s fins in the future and focus on the more traditional fin geometries. The whole simulation method itself that is developed in this thesis is fully capable of studying the effect of VG:s as well. In figure 4.3 the chosen slit geometry is being illustrated. This particular x-slit fin geometry was chosen as the Case 2 for the simulations in this thesis because of its high difficulty level of the meshing procedure. It is based on a composite fin studied by Wu et al. (2014).

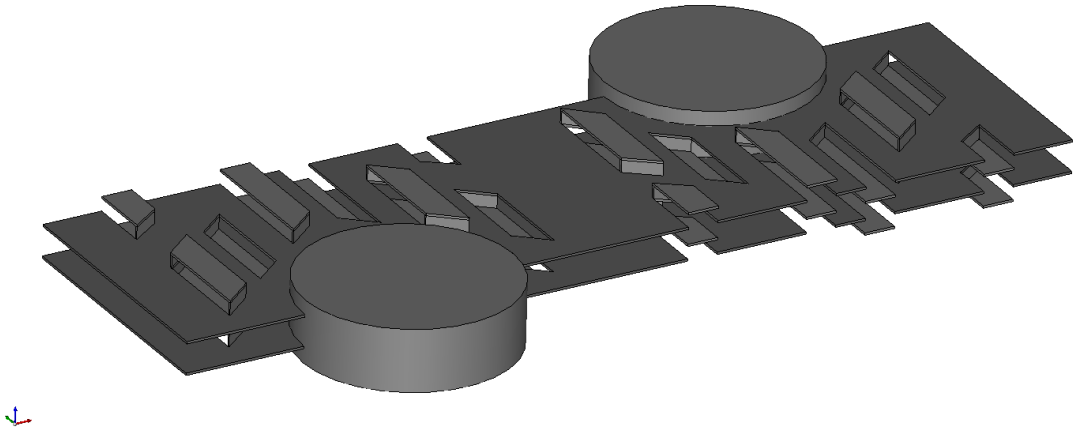
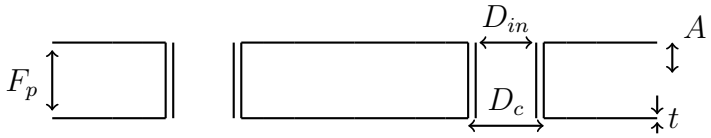


***Figure 4.3*** *The shape of the fin for the x-slit fin case*

These two cases and the geometry details used in this thesis are illustrated in figure 4.4 and the dimensions are listed in table 4.1.

Case 1:




Case 2:

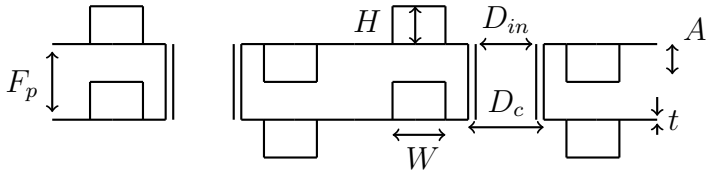



***Figure 4.4*** *The computational domain for the model validation case, created with Salome*

In figure 4.5 an illustration of the longitudinal and transverse pitch is shown as well as the dimensions of the inlet and outlet free stream regions which are used to obtain more stable solutions and avoid oscillations in the fluid domain.

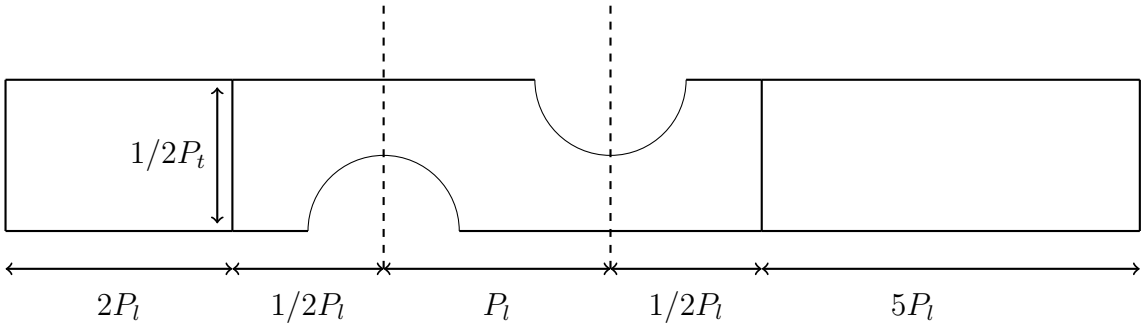



***Figure 4.5*** *Illustration of the transverse and longitudal fin pitch and the inlet and outlet dimensions*

From the table 4.1 it can be seen that neither of the fins have any corrugation in them, in other words the amplitude of the corrugation $A = 0$. All the dimension, for example the diameter of the tubes or fin pitch, are different with these two models. Comparison of these models is still possible to make with the non-dimensional efficiency parameters explained in 2.2.

| Geometry dimensions | | |
|---|---|---|
| Case number | 1 | 2 |
| Fin name | Plane | X-slit |
| Fin thickness (t) | 0.13mm | 0.15mm |
| Fin pitch ($F_p$) | 2.24mm | 2mm |
| Fin collar diameter ($D_c$) | 10.23mm | 13.5mm |
| Amplitude of corrugation (A) | 0mm | 0mm |
| Height of the slit (H) | 0mm | 0.85mm |
| Width of the slit (H) | 0mm | 1.93mm |
| Longitudinal pitch ($P_l$) | 22mm | 28.84mm |
| Transverse pitch ($P_t$) | 25.4mm | 33.3mm |

***Table 4.1*** *Geometry dimensions of the modelled fin types*

It should still be noted that the fluid structures in the air are not independent of these parameters and no small detail comparison should be made between different fin shapes if other parameters are not kept constant.

### 4.1.2 Fluid domains

As said before, the geometries created in this thesis were done by using scripts that are loaded into Salome to create the geometry. The first geometry is the model validation case that is used to compare the results with the experiments to ensure that the computational domain gives correct solutions. In figure 4.6 the whole flow medium is illustrated. It should be mentioned here that this kind of flow domain does not take into account the entry and exit effect of the heat exchanger because the edge of the fin is not modelled. Therefore lower heat exchange and pressure drop characteristics are estimated with this model compared to the one where the fins are modelled correctly and the flow domain extends half way to the fin as illustrated in subsection 4.2.2.

Between the inlet of the domain and the start of the fin, a inlet free stream region is modelled for more realistic flow and for deleting the numerically too perfect inlet conditions. Same things are modelled between the end of the fin and the outlet to avoid oscillations in the outlet values and to delete the interference of the numerical outlet conditions effect on the heat transfer. This region is called outlet free stream and it should be noted here that both left and right side of the inlet- and outlet free stream is set to wall with a slip boundary condition as illustrated in figure 4.6. This region is also adiabatic which means that no heat is transferred through the surfaces in these free stream regions. The other boundary conditions in the figure are opened up in table 4.2 for the model validation case created completely with Salome.

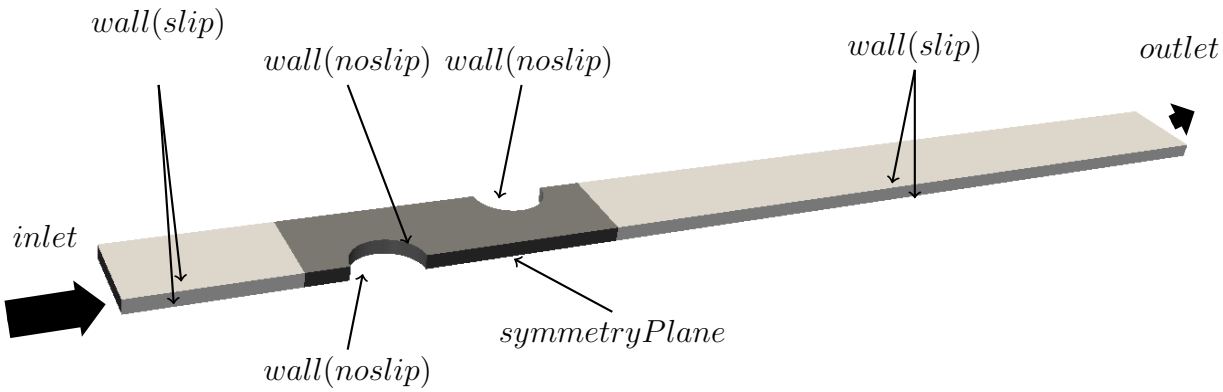



***Figure 4.6*** *The computational domain for the model validation case, created with Salome*

As was mentioned the previous model in figure 4.6 did not include the effect of the entry and exit effect that the fin has due to the difficult meshing procedure needed for the structured mesh creation in Salome. This is why another method was used for more realistic behavior and the flow medium was created with snappyHexMesh and is illustrated in figure 4.7. In this model the flow medium starts and ends from the middle of the fin as illustrated in figure 4.1.

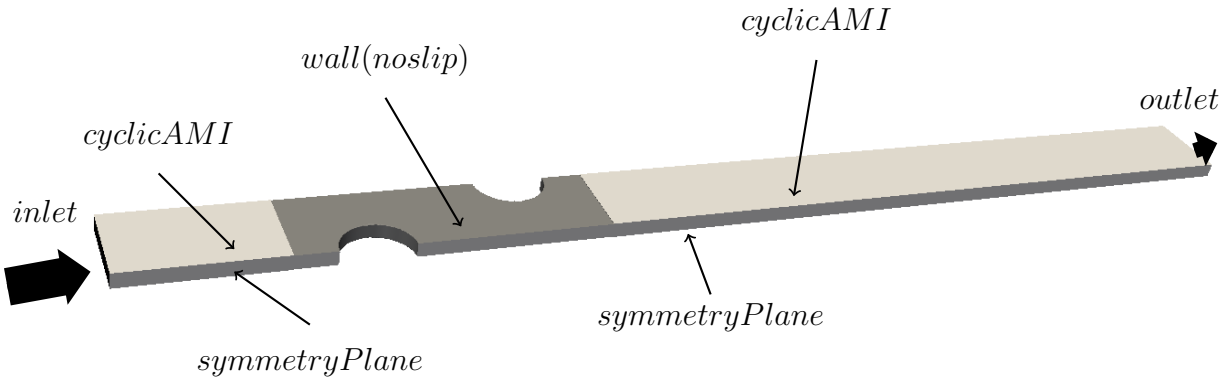



***Figure 4.7*** *The computational domain for the model validation case, created with snappyHexMesh*

The most important difference between these models is that the free stream regions are build in a completely different way. On the structured mesh the whole free stream region is limited with an adiabatic slip wall boundary condition which means that the temperature of the air does not change and the velocity profile stays constant due to non existing viscous effects at the walls. The boundaries are still set as walls which means that they have an effect on the recirculation zone on the outlet free stream region because the boundary condition prevents it from forming naturally. On the unstructured mesh the same effects are created with more realistic boundary condition and the symmetry and cyclic boundaries are set as realistically as possible.

Next the model and the regions that are left after the shape of the fin in figure 4.3 has been cut out of the background mesh explained later in 4.2.2.

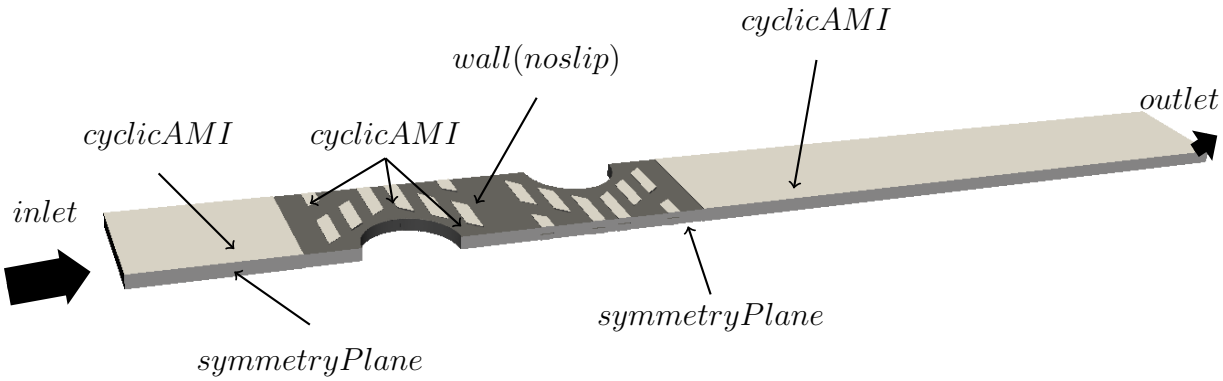



***Figure 4.8*** *The computational domain for the model validation case, created with snappyHexMesh*

It can be clearly seen from the figure 4.8 that meshing the holes perforated in the fin would cause unreasonable amount of work hours and is therefore beneficial to be meshed with an automatic meshing tool.

### 4.1.3 Boundary conditions

For a simple illustration of all the Boundary Condition(B.C.) used in the numerical model, they are explained in table 4.2. Some abbreviations used in the table are fV, zG, WF, sP, cAMI which corresponds to fixedValue, zeroGradient, WallFunction, symmetryPlane and cyclicAMI, respectively. Names correspond to the function objects called in the directory system in OpenFOAM. FixedValue, zeroGradient and symmetryPlane can be said to be self explanatory and will therefore not be paid any attention here. WallFunction on the other hand is a boundary condition that turns on if the $y^+$ value increases above 10 but for this thesis the $y^+$ value is

kept under 1 for all the simulations and therefore the wall functions are not used and the whole flow field is resolved.

| Boundary conditions | | | | | |
|---|---|---|---|---|---|
| Field | k | omega | p | T | U |
| inlet | fV (0.00135-0.576) | fV (0.01-1847) | zG | fV (278K) | fV (0.3-6.2 m/s) |
| outlet | zG | zG | fV 101325Pa | iO | zG |
| wall(slip) | kqrWF | omegaWF | zG | zG | fV (0.0.0) |
| wall(noslip) | kqrWF | omegaWF | zG | fV (333K) | zG |
| symmetryPlane | sP | sP | sP | sP | sP |
| cyclicAIM | cAMI | cAMI | cAMI | cAMI | cAMI |

***Table 4.2*** *Boundary conditions used for different patches along the whole Reynolds spectrum*

The cyclicAMI (Arbitrary Mesh Interface) on the other hand is a bit of a more sophisticated version of the traditional cyclic B.C. In cyclicAMI, for the patches that are linked together it is not necessary to have the same element count and location on the corresponding patches. CyclicAMI can interpolate with two arbitrary patches of size and location. This is crucial for the unstructured mesh cases because the cell number on the corresponding faces is not identical because of the meshing algorithm applied in snappyHexMesh. Difference on these boundaries was found to be the size of few hundreds of cells on the mesh that had around 5 million cells overall. Sometimes the matching of these arbitrary mesh patches is not perfect and can cause the calculation to crash. A solution to this problem was to use a $lowWeightCorrection$ 0.2 in the *boundary* declaration of the boundary condition which changes faces with lower matching than the value specified into zeroGradient. This 20% limit was seen to be a good value and the number of changed faces was then from two to few hundreds with the mesh sizes of few millions.

## 4.2 Mesh

For the simulations in this thesis, two different types of meshes were created. These meshes were created with two fundamentally different *meshing strategies*. The first

strategy was to make fully *structured* meshes with Salome "by hand" to see how many cells would be needed for grid independence and what is the minimum number of cells that needs to be used to achieve accurate results in this type of heat exchanger simulations. In subsection 4.2.1 the procedure of making structured meshes with Salome is explained. As will be seen later in this section, structured meshes are only feasible for simple fin geometries. Especially producing three dimensional unsymmetrical shapes such as slits and louver fins, structured meshes are impossible to create, or would require endless number of hours to be handcrafted. This is why another meshing strategy was created. It is capable of creating meshes around arbitrary fin shapes. These *unstructured* meshes were created with a tool that comes pre-compiled with the OpenFOAM-package. This notorious software is avoided by many users due to its complexity and the long learning curve required to be climbed before good mesh quality can be achieved. In subsection 4.2.2, all the steps required for making meshes with snappyHexMesh for fin-and-tube heat exchangers are illustrated and important features are emphasized.

### 4.2.1 Structured

In figure 4.9 the structured mesh that was created for the Case 1 in Salome is illustrated. The mesh was handcrafted in a way that after the geometry was created with the python script presented earlier, all the faces that define the geometry where flagged as a "Quadrangle face". This makes sure that the meshing tool in the next step will make explicitly hexahedral cells on these faces. The 3D, 2D and 1D algorithms that were used for meshing were Hexa3D, Quandrangle2D and Regular1D, respectively. Then every side of all the faces were defined with 1D sub-mesh algorithm called "wire discretization" and the sub-hypothesis used was "number of segments" for all the sides in the plane that is illustrated in figures 4.9.

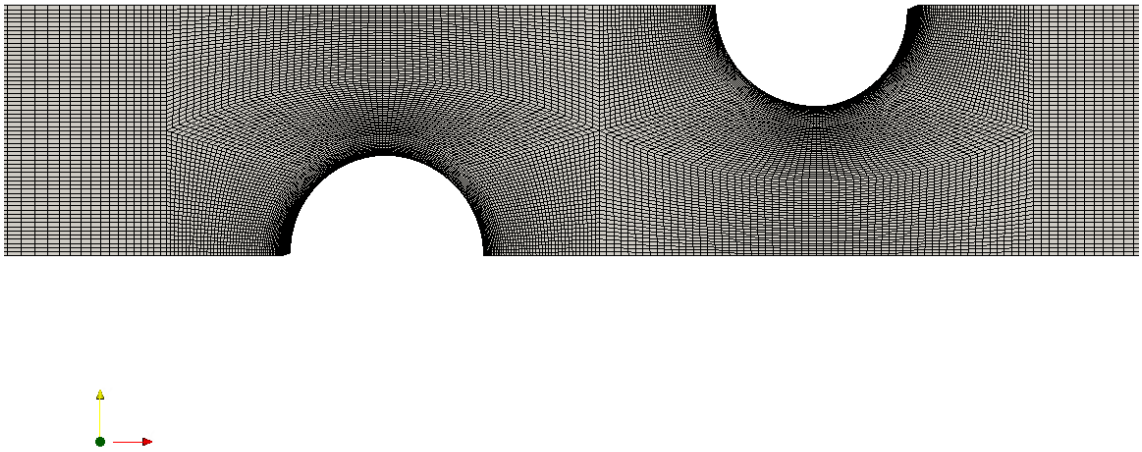


***Figure 4.9*** *The structured mesh created for the model validation case*

As can be seen from the figure, with this type of manual guidance meshing tool, all the boundary layers can be defined individually and explicitly as an engineer would want them to be. This method is highly time-consuming but the result of the mesh will always be in consistent with of the actions made most recently. This way the different areas of the mesh, can be fixed separately and changes made in another parts of the model does not effect the other parts. This method works reasonably well for simple geometries but when some inner features are required inside the mesh some other strategies need to be developed.

For the height of the model a sub-hypothesis "number of segments" was used as can be seen from figure 4.10, but this time the distribution was defined as "Distribution with analytic density" and a second order linear equation was developed to describe the required density distribution of the cells. This way a nice boundary layer refinement was created on the surface of the fins.

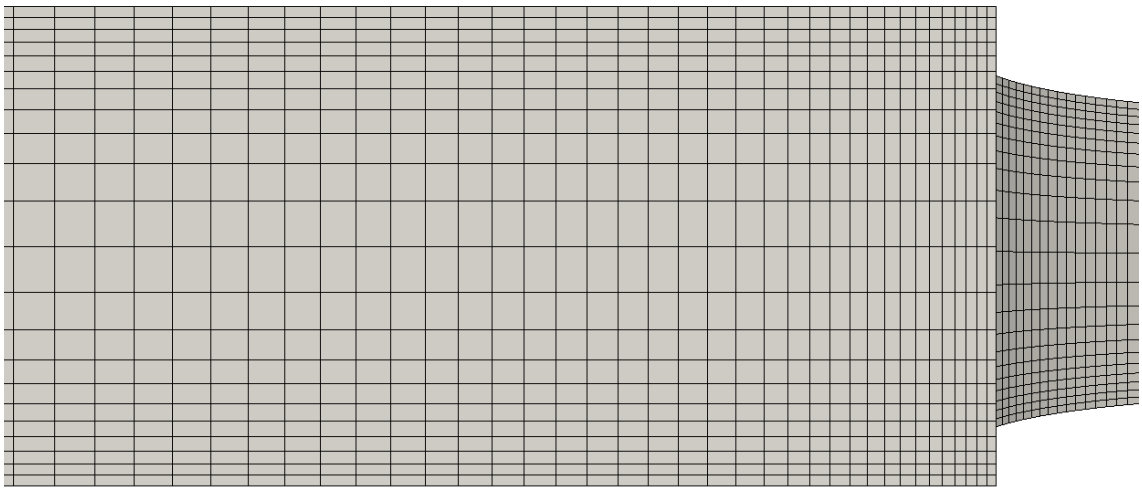


***Figure 4.10*** *The boundary layer refinement for the structured mesh*

### 4.2.2 Unstructured

When more arbitrary geometries are required to be meshed, the previously introduced structured meshing strategy is no longer efficient. This is why a more automated meshing tool was needed and currently there exists only two or three generic software on the open source market. Two of these are cfMesh, developed by Creative Fields Ltd, and snappyHexMesh which comes pre-installed with the OpenFOAM package. Third one is called swiftblock, which is an auxiliary third party tool for another open source tool called Blender. This option was not considered due to the late discovery of the software by the author.

The workflow in these other two meshing tools is completely different. In cfMesh, a surface file is given to the software which is then meshed completely with the set of parameters provided to the algorithm. In no point does the user have an option to choose which side of the model is actually the mesh and what is the body of the geometry. In other words, cfMesh requires a manifold stl-file to function. An

attempt was made with Salome and with another open source CAD-software called Freecad to create a rectangle box and then perform a cutting operation with the fin shape. So that what would be left in the model would be the air that needs to be meshed. Unfortunately author was unsuccessful in doing so. Both CAD-software's crashed every time an cutting operation attempt was made. This is why no suitable stl-file was possible to be made and therefore author was forced to choose to use snappyHexMesh.

In snappyHexMesh the workflow is fundamentally different compared to cfMesh. In snappy, the stl-file that includes the geometry is given to the software and then the user defines that will the mesh be created in- or outside the surface-file. In the first stage the background mesh is provided to the software and the geometry is cut out of it. In this case, a background was made with blockMesh, which is a primitive meshing tool for creation of simple block shapes that are meshed with hexahedral cells. In figure 4.11 an illustration of the background mesh and the stl-file is seen.

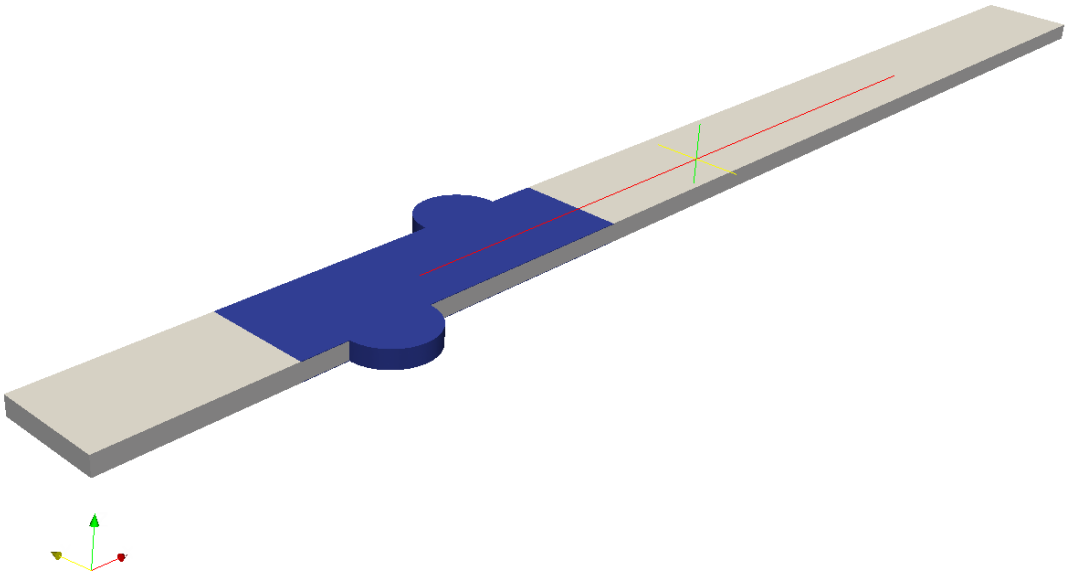


***Figure 4.11*** *Background mesh and the stl-file*

The background mesh is a completely structured mesh, like the ones created with Salome, but it does not include the tubes at all. The mesh is created in a way that it is coarser at the begin and end of the mesh (where the surfaces 0-3-7-4 and 1-2-6-5 are located). And it then refines the mesh closer to the fin-shape region(surfaces 8-9-11-10 and 12-13-15-14). This way the refinements can be adjusted to the regions where they are needed the most. This of course leads to a smaller count of cells required for the simulation. On the fin region, a raise in the background mesh can be created as can be seen from figure 4.12, to be able to mesh all the asymmetric geometries of the fin as will be needed for asymmetric fin shapes for example Herringbone or Sine-wave fins.

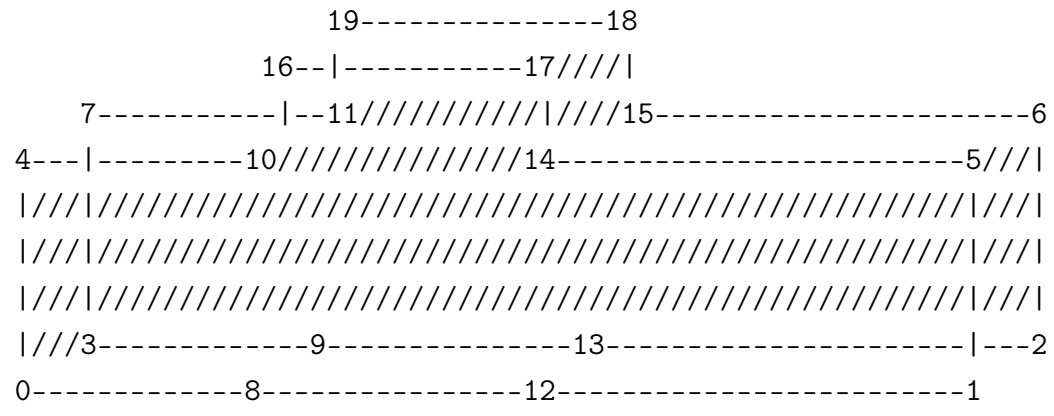



***Figure 4.12*** *Illustration of the background mesh created with blockMesh*

In the workflow of creating the mesh with snappyHexMesh there exists three different stages. In the first phase of the meshing process, snappy cuts the stl-file out of the background mesh and leaves the cells that have less than half of their volume on the surface file side. This way a *castellated mesh* is created and a close up of the formed mesh is shown in figure 4.13. The whole process is controlled with a set of parameters defined in a snappyHexMeshDict which is located in the system-directory. Certain number of aspects are important to take into consideration in every step when meshing with snappy. In the phase one, it is important that the background mesh is fine enough that the castellated mesh will be created with sufficiently enough number of cells and the geometry is visible. The phase one is controlled with a set of parameters which are shown below and important ones are highlighted with comments.

```
geometry
{
    model_validation.stl //This is the name of the given surface-stl-file
```

```
        {
            type triSurfaceMesh;
            name finsandtubes; //this is the name of the surface after the
                               //cutting process
        }
};

castellatedMeshControls
{
    maxLocalCells 100000000;  //This needs to be high enough,
    maxGlobalCells 100000000; //otherwise the algorithm stops
                              //in the middle of the process
    minRefinementCells 0;
    nCellsBetweenLevels 3;

    features
    ( //This is a more detailed surface feature file
       { //which is created with a surfaceFeatureExtract tool
   file  "model_validation.extendedFeatureEdgeMesh";
   levels ((0.0 0));
   }
    );

    refinementSurfaces
    {
       finsandtubes
                { //Here a lowest and highest refinement level is defined.
                level (1  1);  //For this case level 1 was a good compromise
        patchInfo {    //with surface recognition and cell count.
             type wall;
                  }
        }
    }

    resolveFeatureAngle 10;
    refinementRegions
```

```
    {//Here a number of refinement regions can be defined
    }
  locationInMesh (0.002 0.001 0.001 ); //This defines a point in the
  allowFreeStandingZoneFaces false;    //geometry that will be inside
}                                      //the created mesh.
```

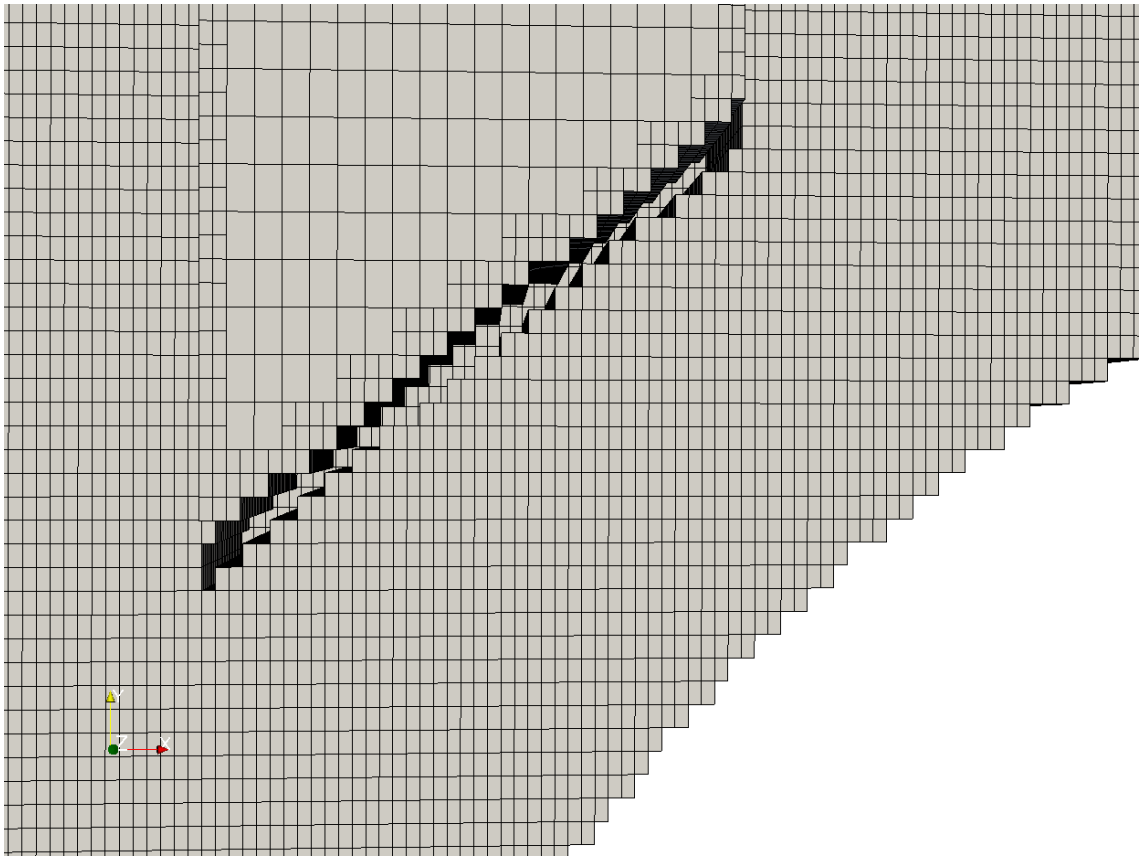


***Figure 4.13*** *The castellated mesh after the first phase of the snappyHexMesh meshing procedure*

Refinement level was set to be minimum of 1 and a maximum of 1, this means that every cell that is located on the surface will be divided in half in all three direction. This means that every cell on the surface transforms into 8 smaller cells. If the maximum level would have been chosen to be on the level 2, every level 1 cell would then be again divided into 8 cells and so on. This of course leads to a great increase in the cell count.

In the phase number 2, the shape of the mesh will be "snapped" to match the geometry of the surface file. This means that the cell corners will be moved closer to the surface and the rest of the mesh is smoothed and bad cells are removed. The

phase two is controlled with a set of parameters which are shown and important ones are highlighted.

```
snapControls
{
    nSmoothPatch 15; //How many times the cell points are moved
    tolerance 0.8; //How far away the points are moved on the
             // surface True distance is this factor times
             // local maximum edge length
    nSolveIter 20;
    nRelaxIter 5;
    nFeatureSnapIter 15;
    implicitFeatureSnap false; //Self detection of surface features
    explicitFeatureSnap true; //Features are provided with a file
    multiRegionFeatureSnap true;
}
```

The mesh created with these parameters can be seen figure 4.14. The geometry created with snappyHexMesh is not exactly following the shape of the surface file that was provided but a reasonable recreation is achieved.

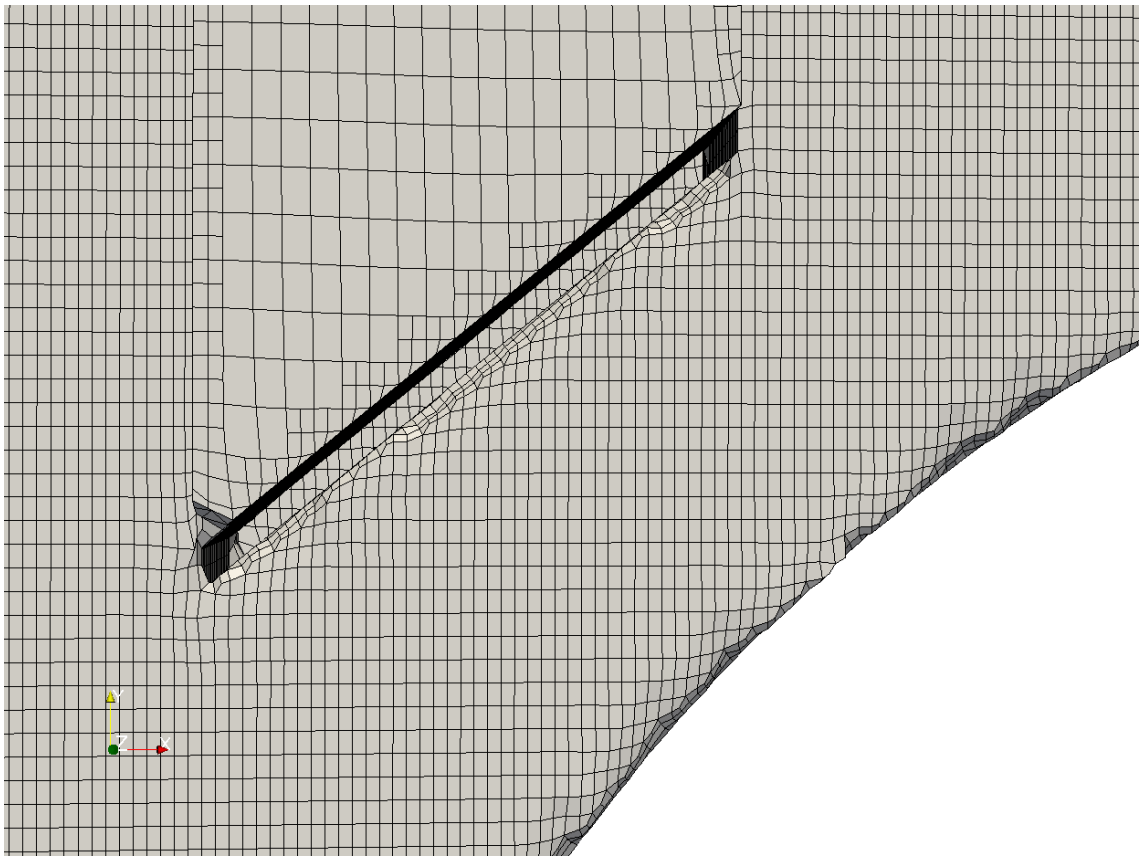


***Figure 4.14*** *The snapped mesh after the second phase of the snappyHexMesh meshing procedure*

It was noted that it is not easy to run this part of the meshing process. Many problems raised with the detection of round surfaces (the tubes) and sharp 90-degree angles. And usually only one of them at a time was successfully snapped. As can be seen from figure 4.14 some smoothing of the corners was still occurring and some bad cells were created in the mesh even after months of testing and comparing different parameter setups. Due to limited time provided for the thesis and uncertainty of ever reaching a perfect setup, a set of parameters was chosen and then used for all cases. The reality of course does never have sharp angles between the fin and the tube so some smoothing in the flow medium due to meshing process can be said to be acceptable. When thinking about heat transfer the importance of the flow field near the walls can not be neglected. Therefore a boundary layer refinements were decided to be used instead of wall functions in the wall regions. In snappyhexmesh, a last phase in the meshing process creates these layers. A number of parameters were again used which are:

```
addLayersControls
{
    layers
    {
        finsandtubes //Name of the surface afted snapping
          {
            nSurfaceLayers 6; //number of layers wanted
      }
    }
   relativeSizes true; //Are the sizes relative to cells size
    expansionRatio 1.3; //Expansion ratio of the layers
    finalLayerThickness 0.6; //Thickness of the last layer
    minThickness 0.000001;//When this is set low enough,
     //it does not effect the layer addition process
    nGrow 0;
    featureAngle 270; //Largest angle that the boundary layer
    //will be expanded over
    slipFeatureAngle 20;
    nRelaxIter 5;
    nSmoothSurfaceNormals 5;
    nSmoothNormals 3;
    nSmoothThickness 15;
    maxFaceThicknessRatio 0.5;
    maxThicknessToMedialRatio 1000;
    minMedianAxisAngle 300;
nMedialAxisIter 20;
    nBufferCellsNoExtrude 0;
    nLayerIter 50;
    nRelaxedIter 20;
    additionalReporting true;
}
```

Many problems were faced in this third phase of using snappyHexMesh. One reason seems to be that the meshing process does not know how to proceed only from the phase 2 to phase 3, in other words only through the layer addition part. When used in this manner, the meshing tool pretends to be creating the layers but never actually saves them in to the files. This is why the whole meshing process needs to

be done again every time a new boundary layer version needs to made. One meshing run takes around 2 hours for the mesh with approx. 2.5 million cells and therefore doing studies with different parameters leads to many hours of waiting. A smaller meshing model with enough features to represent the real case is essential in this process. In figure 4.15 an illustration of the same mesh after the last phase of the meshing process is shown.

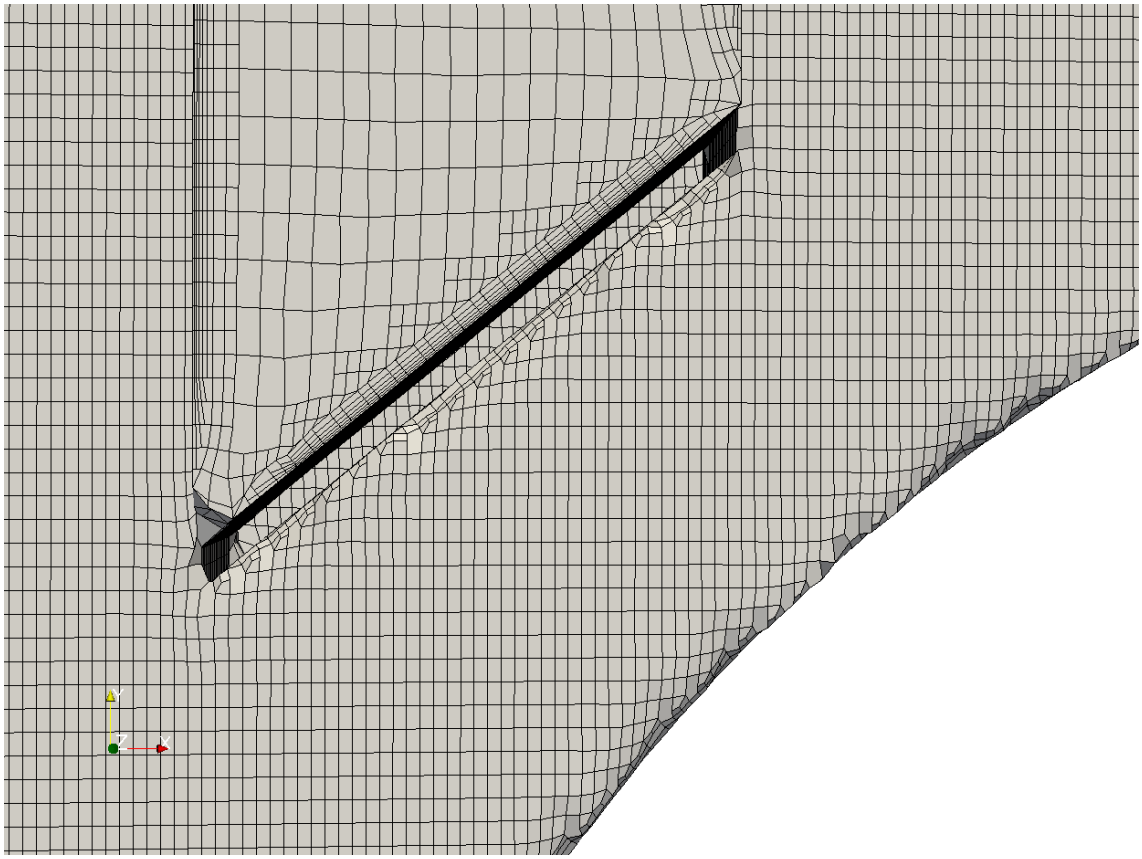


***Figure 4.15*** *The final mesh after the third phase of the snappyHexMesh meshing procedure*

In figure 4.16 an example of the boundary layers on the inlet part of the flow medium is shown. The algorithm recognises the shapes around at the inlet of the fin where the first boundary layers starts to form but some problems can be found at the corners in the region where the fin is connected to the tube.

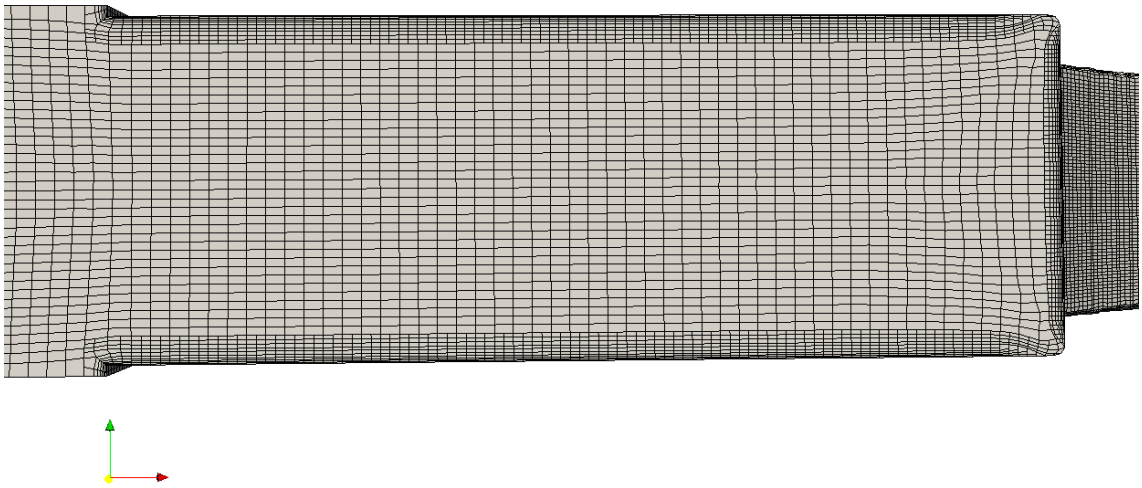


***Figure 4.16*** *Mesh details of the model validation case meshed with snappyHexMesh*

In figure 4.17 an illustration of the problems with the boundary layer addition in the corners can be seen. The reasons for the bad quality can be said to accumulate through the phases of the meshing process and no clear reason or solution to these problems were found in this thesis. Some of the reasons are smoothed shapes with non orthogonal cells that are difficult for the snappy boundary layer addition algorithm and another big reason is the deviation of the angles of shapes in the mesh.

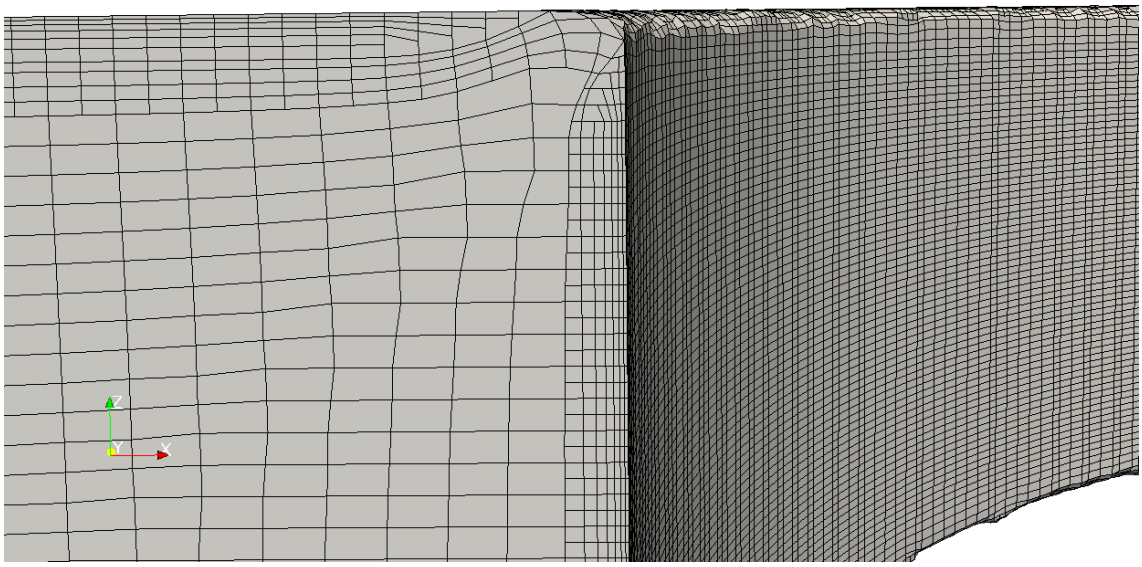


***Figure 4.17*** *Mesh details of the model validation case meshed with snappyHexMesh*

The quality of the stl-file was also found to be one important factor in the search of a good mesh. To increase the quality of the final mesh product, the surface file that was given as the boundary of the mesh for snappy was meshed with surface tetra cells. This means that the shapes were described by a greater number of points and is therefore easier for the algorithm to follow. A gentle overkill was done in this process to ensure a good shape recognition. Overall it can be said that meshing with snappyHexMesh has a huge potential for meshing arbitrary shapes with hexahedral cells. The algorithms capability of shape recognition and layer adding features are still not on a level that good quality meshes could be created in a short one day time period as often needed in the engineering offices. A great number of parameter iteration is always needed if the author lacks experience with snappeHexMesh.

### 4.2.3 Grid independence

To ensure that the mesh used for the simulations is independent of the size of the cells a number of different mesh sizes were tested. The mass flow averaged

temperature in the outlet was chosen as a criteria for the independence variable due to its importance to the final results. It is proportional to the j-factor calculated for every fin shape. In table 4.3 is the summary of the grid independence study done for all the three simulations. Three different meshes were tested and the converged outlet temperature is shown for each mesh size.

| Grid independence study | | | |
|---|---|---|---|
| number of cells | 0.27mil | 0.5mil | 1.2mil |
| Temperature for Case 1 structured | 300K | 304K | 304.5K |
| Difference | 17% | 1.9% | 0% |
| number of cells | 1mil | 2.5mil | 5mil |
| Temperature for Case 1 unstructured | 301.880K | 302.381K | 302.387K |
| Difference | 0.92% | 0.1% | 0% |
| number of cells | 1.7mil | 3.7mil | 6.8mil |
| Temperature for Case 2 unstructured | 324.9K | 324.71K | 324.21K |
| Difference | 1.45% | 0.9% | 0% |

***Table 4.3*** *Grid Independence study with Reynolds number of 7000 for different meshes*

First the structured mesh for Case 1 was studied, which represents the minimum number of elements needed for this kind of heat exchanger simulation. A fairly small mesh with only 0.5 million cells with a difference to the 1.2 million cells of only 1.9% was chosen. For the unstructured mesh of the Case 1 a bit finer mesh with 2.5 million cells was chosen for the simulations. This mesh has a difference of 0.1% respect to the one calculated with 5 million cells. Another smaller mesh with 1 million cells was also compared with the 5 million one, but this was not used due to the bad quality of the mesh cells created by the snappyHexMesh algorithm.

For the Case 2 another three meshes were compared. For the smallest mesh, with 1.7 million cells, snappyHexMesh was not able to create the boundary layer at all. This is due to the bad quality of the cells after the snapping phase. This is why a 3.7 million cells mesh was used instead even though the difference in the outlet temperature differs only 1.45 percent from the largest mesh with 6.8 million cells. The difference for the 3.7 million cells was found to be 0.9%, which can be said to be acceptable in respect to the accuracy of this calculation process.

One interesting aspect with the high resolution meshes was that the temperature value at the outlet was found to fluctuate and never to converge into one value. This could be because of the resolved structures in the flow field with the finer grid. This is why an average of the last few hundred values were taken for the final results shown in the table before.

It should be emphasized here that a vast number of problems were encountered with snappyHexMesh when down-scaling the number of elements in the mesh for the grid independence study. If the coarsement was too high and no node points in the background mesh were located inside the geometry, the meshing process often crashed. Another problem was encountered when snappyHexMesh moves to the next phase, the snapping phase. If the tolerance was set too low and the sizes of the cells were too big, the snapping process was not successful, which leads to failure in the boundary layer adding step. Therefore it can be said that the limiting criteria for using a certain number of cells is not the poorly resolved field of too coarse mesh, but more the amount of cells needed for the snappyHexMesh algorithm to finish the mesh successfully.

### 4.2.4 $y^+$ -study

No thorough $y^+$ studies were done for the meshes in this thesis because no wall functions were used. It is known that $k-\omega$ SST does not use wall functions if the $y^+$ value at the wall is kept below 1. This is why all the used meshes for the simulations were checked with the highest inlet velocity that the values are well below this limit. The average $y^+$ values for Case 1 structured, Case 1 unstructured and Case 2 with Reynolds number of 7000 were 0.4, 0.029 and 0.05, respectively.

### 4.2.5 Discretization Scheme

For discretization schemes in this thesis a wide range of choices was available in OpenFOAM. Because the direction of the flow in the fluid domain is mainly unidirectional, a second order upwind scheme should give the best results. Second order upwind scheme means that the value calculated in a point in the flow medium is calculated from the points that are located upstream from the referred point. In this thesis the velocity U and energy e fields were calculated with using second order upwind scheme whereas for turbulence quantities a first order upwind scheme was used and for all the others Gauss linear scheme was selected.

### 4.2.6 Measure of Convergence

For a measure of convergence of the computational model the residuals should be monitored. Residual is the difference between the result of the latest time step and the one before the latest. This means that when the residuals of the solution are low, it can be said for the specific mathematical model that the field that is being calculated with those boundary conditions has reached a solution in the frame of that set up. However if one monitors only the residuals this can lead to stop the solution before the simulation is actually finished. Or if the residuals never go below a certain point, one can not say that has the calculation reached convergence or not. This is why it is a good habit to monitor some crucial parameter or a variable that is essential for the final result of the simulation. When this reaches a value, it can be said that at least with these settings the field has find a convergence. In this thesis the mass flow averaged temperature at the outlet was monitored with a script shown in appendix B. In figure 4.18 is an illustration which shows how the residuals looked like for the case 1 with Reynolds number 7000. An established unwritten rule in CFD scene has been that a $10^{-5}$ convergence limit is widely used in various different CFD-studies for a sign of convergence. For the simulations in this thesis this limit was never reached in any of the cases.

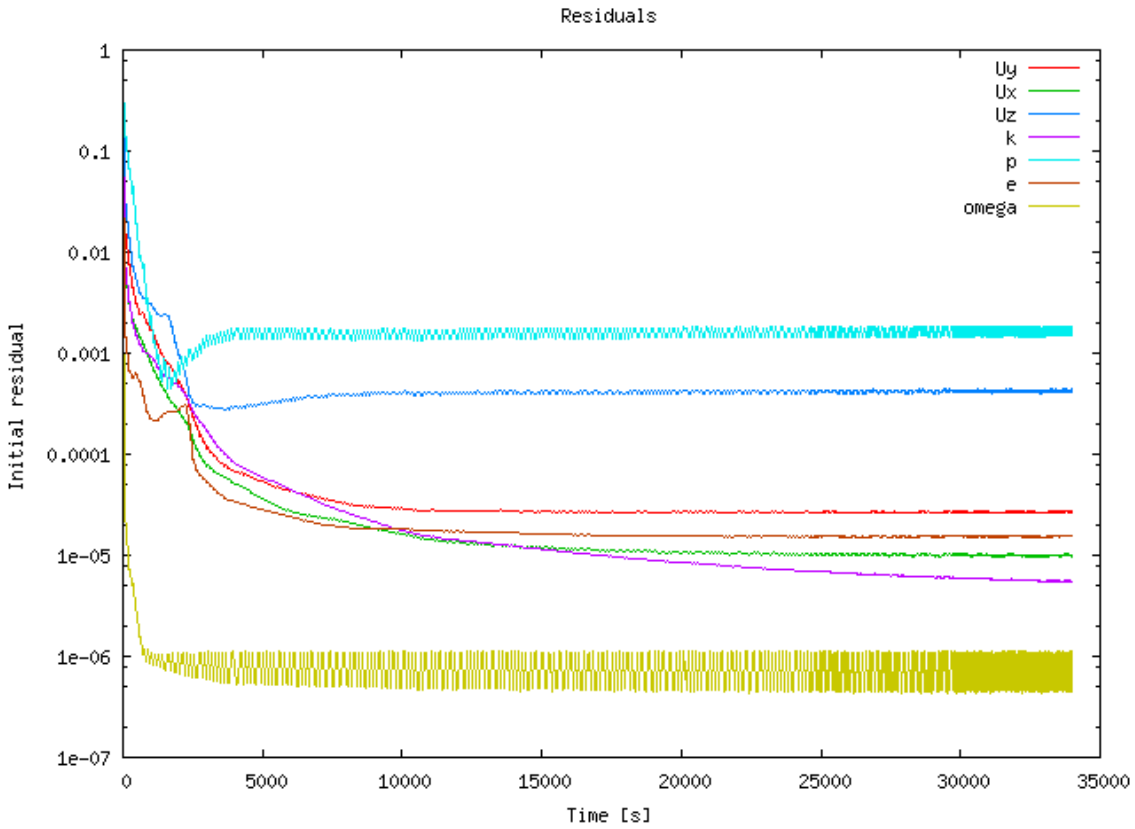



***Figure 4.18*** *Monitor mass weighted temperature at the outlet*

Even though the residual for pressure p, does not sink below 0.001, it can be seen from figure 4.19 that the solution in respect to the temperature at the outlet, which is used for the calculation of the j-factor, reaches a converged value finally after 20000 time step. Therefore it can be concluded that the solution has reached convergence and the simulation is finished.

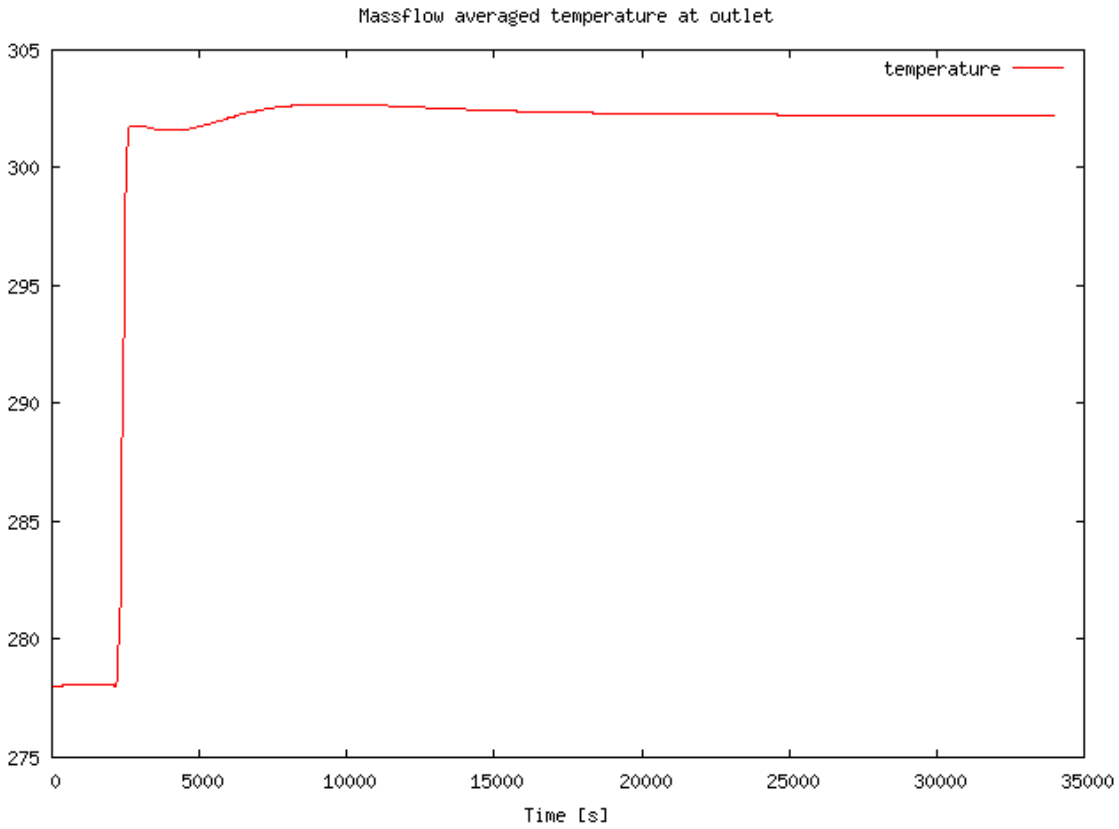



***Figure 4.19*** *Monitored mass weighted temperature at the outlet*

It was notable that the residuals and monitored values for the structured mesh fluctuated more than the ones from unstructured mesh. Reasons to this can be the boundary conditions on the free stream regions, where the other one has walls and the other one does not. Another reason could be that since unstructured meshes had a lot more elements, more numerical diffusion will be created that dampens these fluctuations.

Overall it was seen that the highest velocities, 5.4 m/s and 6.2 m/s, did not converge as easily as the lower ones. This is probably because of the unsteady nature of the flow field in this kind of high Reynolds number flows. 5

# 5. TURBULENT FLOW

Turbulent flow is used in practical heat exchanger applications to enhance heat transfer. Turbulent flow still always comes with a cost, which is the increased pressure drop in the fluid medium. In this project, the flow is modelled with turbulence models due to the lack of super computers and aim to achieve an efficient way to study heat exchangers in the future, forces us to use computationally cheaper approaches. In this chapter, some main characteristics of turbulent flow and its importance to the flow calculation will be introduced. Then a review of different models available in OpenFOAM will be discussed in order to make a reasoned decision of the model to use for the application of this thesis. Finally we look in to more detail to few potential turbulence models which are used widely in industry and has been proven to be the best compromise between accuracy and computational cost.

## 5.1 Turbulence characteristics

Throughout the years, scientists and engineers have developed analytical solutions to various different flow cases to answer the questions regarding the behaviour of the flow. These are usually fundamentally simplified cases with mathematically describable features. One of these main features is that the flow needs to be *laminar*, meaning that the flow is smooth and steady (Frank M, 2003). In the real world the flow almost never fulfils these requirements and therefore the flow field can not be solved by analytical means and is then called *turbulent*. There exists no precise definition of what is turbulent flow and a clear separation between laminar and turbulent flow, cannot be drawn. But still according to Pope (2000), Tennekes and Lumley (1972) and Davidson (2015), turbulent flow and turbulence itself can be said to have certain universal characteristics such as:

1. **Irregularity**. This characteristic emphasizes the nature of turbulence to be irregular, random or even chaotic, meaning that prediction of turbulent behaviour is, if not possible, a very challenging task. Despite of its irregular

nature, it can be studied with deterministic approaches and Navier-Stokes equations still describe the full nature of the flow.

2. **Diffusivity**. Turbulent flow is more diffusive than laminar one. Increased diffusivity can be said to be the most important feature of turbulent flow since it increases the transfer of momentum, heat and mass. This can be used in practical applications for example to delay the separation of the flow from the body to increase the possible angle of attack for airfoils and to increase the heat exchange in heat exchangers. It is also a source of resistance of the flow in ducts and pipes.

3. **Large Reynolds number**. Turbulent flow always occurs in flow situations with high *Reynolds* numbers. This can be said to be over $Re_D \geq 2300$ for the flow inside a circular tube or $Re_x \geq 500000$ for the flow over a plate. When the Reynolds number increases the instabilities in the flow increase with it. This complex interaction in the momentum equation between the viscous and inertial terms is one of the unsolved problems in the world still in 2015.

4. **Three-Dimensional**. Turbulence is always a three dimensional phenomenon. This is mainly because the main feature that enhances turbulence is vortex stretching. It is the main mechanism of transforming turbulent energy to smaller scales.

5. **Dissipation** In a turbulent flow, the big eddies are created by the mean flow and their size can be as large as the length scale of the mean flow. This turbulent energy is then transferred to smaller, medium sized eddies and again to smaller and smaller eddies, until the smallest eddies are transferred into thermal energy. This process is called the *cascade process*, and it explains why turbulent flow is dissipative in its nature.

6. **Continuum**. The length scales of the turbulent eddies can be considered much bigger than the molecular length scales as explained in section 3.1.

## 5.2 The main approaches of making turbulent flow simulations

As discussed earlier the wide range of turbulent eddies with different length and time scales have an impact to the overall flow field in a way that is both highly complex in its all three dimensions as well as time depending. It is clear therefore that all kind

of time-averaging will loose some of the features of the original turbulence. Since it is an important feature of the flow, that can not be neglected. It then always comes to the decision of how much should be calculated and how much should be modelled in a rally for the best solutions achievable with the computational resources available. The approaches can be separated in to three fundamentally different approaches which will be discussed next.

### 5.2.1 Turbulence modelling of Reynolds-averaged Navier-Stokes(RAN equations

Only global changes are considered important and therefore only mean values of the flow are calculated. Different turbulence models, for example $k - \epsilon$ or Reynolds Stress Models (RSM)(see 5.3.3) are used to link turbulent fluctuations influence on the mean flow. This method can never achieve a level of fully realistic flow features but for some engineering tasks it is accurate enough. Computational resource-wise this is by far the cheapest solution and therefore widely spread in the engineering scene. RANS turbulence modelling is also used in this Master's Thesis as more thoroughly explained later in this chapter.

### 5.2.2 Large Eddy Simulation(LES)

The idea in Large Eddy Simulation is to resolve the bigger features, larger eddies, in the flow and model the smaller scales for cheaper computational price but still gaining accurate enough results. A certain filtering system needs to be established for the separation between these two. This also creates a problem called *grey area problem* which stresses the difference in the contact area between simulated bigger eddies and modelled smaller eddies. This can cause a big abrupt change in flow variables such as density or viscosity. LES can be said to be "the tool of the future", due to more accurate solutions and realistically achievable computational resources in the next two decades, for industrial usage.

### 5.2.3 Direct Numerical Simulation (DNS)

In direct numerical simulation, all the scales are calculated, from big to all the way to the smallest Kolmogorov length scales, where the smallest eddies are transformed into energy. The time step will be chosen small enough that the most important

characteristics of turbulent fluctuations are calculated. Of course, all this, is computationally really expensive and is therefore not in the reach of industrial use for years and years to come.

## 5.3 Reynold's Averaged Navier Stokes(RANS)

In this section we go through the process of introducing the fluctuating transient part $v_i'$ and mean flow velocity $\overline{v_i}$ of the turbulent flow velocity and how they affect the governing equations.

### 5.3.1 Time averaged flow properties

If we look at a point in a three dimensional flow medium and plot the velocity component of only one direction, over the time, the plot could look something like in figure 5.1 below.

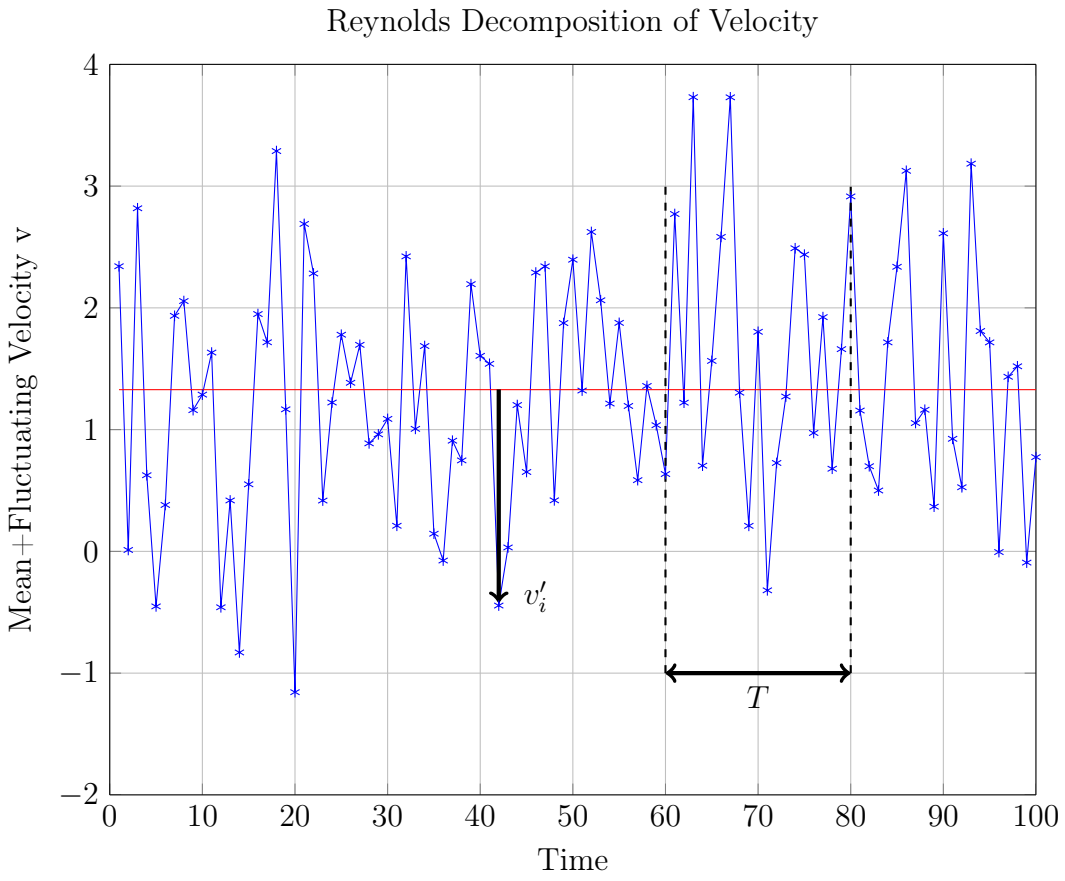



***Figure 5.1*** *Two components of turbulent velocity:* $v_1'$ *is the transient fluctuating component and* $\overline{v_1}$ *is the mean velocity component*

Where we can see the mean velocity in x-direction to be $\overline{v_i}$ and the turbulent fluctuation $v_i'$ around the average. These together, they form the actual transient velocity $v_i = \overline{v_1} + v_1'$. These fluctuation components are caused by the momentum exchange from vertical eddie motions that accelerate the slow moving layers and decelerate the fast moving layers according to the conservation of mass(see 3.1).

Usually for engineering applications, only global changes in fluid properties are the point of interest and therefore time-averaged equations can be used for simulation. As was shown above, the velocity was broken down in to two components, mean and the fluctuating one. Other properties such as pressure and temperature have fluctuating components in all three directions as well. Lets introduce a steady mean component $\overline{\phi}$ and break down the process of how to calculate it for an arbitrary fluid property.

As decomposed for the velocity before, for $\phi$, the decomposition can be made in a similar fashion:

$$\phi_i = \overline{\phi_i} + \phi_i' \tag{5.1}$$

And therefore the mean properties of $\phi$ during the time interval $T$ can be calculated with equation

$$\overline{\phi_i} = \frac{1}{T}\int_{t-\frac{T}{2}}^{t+\frac{T}{2}} \phi_i dt \tag{5.2}$$

and is therefore the time averaged value of the property $\phi$. On the other hand if the fluctuating component $\phi_i'$ is averaged over time:

$$\overline{\phi_i'} = \frac{1}{T}\int_{t-\frac{T}{2}}^{t+\frac{T}{2}} \phi_i dt \equiv 0 \tag{5.3}$$

it is zero as can been seen intuitively from figure 5.1. It is important to notice that the time average of *variance* or *root mean square*(r.m.s) of these fluctuation properties will not equal to zero. Also we must note that second or higher moments between these different properties will not be equal to zero. If we consider the previously introduced property $\phi_i = \overline{\phi_i} + \phi_i'$ and another arbitrary property $\psi_i = \overline{\psi_i} + \psi_i'$ their second moment is defined as

$$\overline{\phi_i'\psi_i'} = \frac{1}{T}\int_{t-\frac{T}{2}}^{t+\frac{T}{2}} \phi_i\psi_i dt \tag{5.4}$$

which is not zero as this is the case for higher order moments also. (Versteeg and Malalasekera, 2007)

### 5.3.2 Time averaged Navier-Stokes equations

When we introduce these time averaged flow properties formulated in subsection 5.3.1 to the governing equations the continuity 3.1 and momentum 3.2 explained in section 3.3 we end up with the time averaged equations for all the turbulence properties. First we go through more in detail how this process is done for the continuity equation and then only the final formulation for momentum and scalar transport equation is illustrated.

Here we also introduce the time dependent part of the continuity equation because turbulence is a time dependent phenomenon. Therefore continuity equation for transient three dimensional(opened up with rules explained in appendix C) flow can be described as

$$\frac{\partial\rho}{\partial t} + \frac{\partial(\rho u)}{\partial x} + \frac{\partial(\rho v)}{\partial y} + \frac{\partial(\rho w)}{\partial z} = 0 \tag{5.5}$$

If we now replace the density $\rho$ and the three Cartesian velocity components $u, v$ and $w$ with the mean component sum of $\overline{\phi}$ and fluctuating component $\phi'$ we will come up with a following formulation for the continuity equation

$$\frac{\partial(\overline{\rho}+\rho')}{\partial t} + \frac{\partial(\rho(\overline{u}+u'))}{\partial x} + \frac{\partial(\rho(\overline{v}+v'))}{\partial y} + \frac{\partial(\rho(\overline{w}+w'))}{\partial z} = 0 \tag{5.6}$$

When we now remember the time averaging rules from equation 5.2 and 5.3 we can conclude that the final form for time averaged continuity equation is as follows

$$\frac{\partial\overline{\rho}}{\partial t} + \frac{\partial(\rho\overline{u})}{\partial x} + \frac{\partial(\rho\overline{v})}{\partial y} + \frac{\partial(\rho\overline{w})}{\partial z} = 0 \tag{5.7}$$

When we repeat the same procedure for the momentum equation 3.2 we conclude that the corresponding equation for x-direction is as follows

$$\frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(\rho\overline{u^2}) + \frac{\partial}{\partial y}(\rho\overline{vu}) + \frac{\partial}{\partial z}(\rho\overline{wu}) = -\frac{\partial \overline{p}}{\partial x} + \mu \cdot [\frac{\partial^2 \overline{u}}{\partial x^2} + \frac{\partial^2 \overline{u}}{\partial y^2} + \frac{\partial^2 \overline{u}}{\partial z^2}] + \underbrace{\frac{\partial}{\partial x}(-\rho\overline{u'^2}) + \frac{\partial}{\partial y}(-\rho\overline{v'u'}) + \frac{\partial}{\partial z}(-\rho\overline{w'u'})}_{\text{new}} \tag{5.8}$$

It is important to notice here that the corresponding equation has the same set of terms as the generic momentum equation and only the highlighted terms are new and are introduced after the time averaging process. Another important thing to notice here is that when we have formulated the time averaged turbulent momentum equation 5.8, we assume, that the fluid density is constant. But as can be guessed this is intuitively not always the case. Common engineering flows often involve situations where density can drastically vary globally, due to temperature variation or high flow speeds, not to mention the turbulent fluctuations previously discussed. Even though, Bradshaw et al. (1981) have shown that in subsonic($Ma < 1$) flow speeds the density fluctuations can be neglected and therefore $\rho$ is considered constant, in local element scale formulation. The new stresses introduced in the momentum equation can be considered to be a new set of stresses that are caused by the turbulent fluctuation and they are named Reynold stresses.

$$\tau_{ij} = -\rho(\overline{u_i' v_j'}) = \begin{Bmatrix} \rho\overline{u'^2} & \rho\overline{v'u'} & \rho\overline{w'u'} \\ \rho\overline{u'v'} & \rho\overline{v'^2} & \rho\overline{w'v'} \\ \rho\overline{u'w'} & \rho\overline{v'w'} & \rho\overline{w'^2} \end{Bmatrix} \tag{5.9}$$

It should be noted here that the reynolds stress tensor 5.9 is symmetric, because $\overline{u'v'} = \overline{v'u'}$, and therefore only 6 new independet unknowns are introduced in the time averaging process.

Now the only one left that does not have an equation is the time averaged arbitrary scalar variable. This can be expressed in a form as follows:

$$\frac{\partial \rho\overline{\phi}}{\partial t} + \mathrm{div}(\rho\overline{\phi u}) = \mathrm{div}(\Gamma_{\overline{\phi}}\mathrm{grad}\overline{\phi}) + \underbrace{\frac{\partial}{\partial x}(-\rho\overline{u'\phi'}) + \frac{\partial}{\partial y}(-\rho\overline{v'\phi'}) + \frac{\partial}{\partial z}(-\rho\overline{w'\phi'})}_{\text{new}} + S_\phi \tag{5.10}$$

where the corresponding terms are the same as in the general transport equation 3.5 and the new ones are the induced turbulent fluctuations.

The previously introduced turbulent terms can be modelled with numerous different ways and the development of these models can be a highly challenging task. A useful way of handling difficult expressions is to give a classification for different turbulent properties. A short introduction to these is found in the appendix D.

### 5.3.3 Model comparison

Next we should consider which turbulence models are available in OpenFOAM and look into more detail of their characteristics, limitations and potential for the use for heat exchanger modelling. In the appendix E on the table 1 is a listing of turbulence models that have a compressible as well as incompressible variety in OpenFOAM, where as table 2 contains the models that have only compressible or a incompressible variety.

### 5.3.4 Turbulence model for the air flow inside fin-and-tube heat exchanger

When we consider the important task of choosing the most suitable turbulence model for the simulation of heat transfer in a fin-and-tube heat exchanger, a certain number of key factors can be pointed out. These factors can be divided in to flow characteristics that need to be simulated correctly for accurate heat transfer prediction.

- Detachment of the flow from the tube profile
- Recirculation zone behind the tubes
- Turbulent heat diffusion due to turbulent fluctuations

These flow characteristics are the same important characteristics that are usually poorly simulated with turbulence models. For example in the most traditional $k-\epsilon$ model, the turbulent kinetic viscosity ( 5) is over predicted near the wall which then leads to lower dissipation of turbulent kinetic energy as would be in reality. This leads to over-prediction of the shear stress especially in adverse pressure gradient flows which the flow around the tube can be considered as. This is usually fixed by introducing a damping function for the calculations near the wall (Davidson, 2015).

Another study has been done by Hansen (2008), where a similar fin-and-tube heat exchanger simulation with OpenFOAM was done. Hansen studied the flow in the same plain fin geometry case that is used in this thesis for model validation. Hansen ran the simulations with *laminar*, $k-\epsilon$ and $k-\omega$ SST models and concluded that at the lower velocities the laminar, a dummy turbulence models, gave the closest heat transfer and pressure predictions from all these three compared to the experiments. Where as for higher velocities $k-\omega$ SST was the best choice. Overall it was concluded that on average $k-\omega$ SST gave the best prediction of these flow variables and it is therefore chosen as the turbulence models for this thesis.

Reasons why $k-\omega$ SST(Shear Stress Transport) should perform better for the flow inside a fin-and-tube heat exchanger is that this model uses the $k-\epsilon$ model in the free stream but switches to $k-\omega$ in the near wall region. This is known to work better for predicting the detachment and attachment of the flow and gives more correct shear stress values in the recirculation regions.

Some other turbulence models of course have been considered to be suitable for the calculations in this thesis. For example Reynold Stress Models(RSM) or Algebraic Reynold Stress Models(ASM) could predict the flow features with higher precision but it is known to be much more computationally expensive and is numerically more unstable than the $k-\omega$ SST. And because almost the whole simulation method developed features in thesis are done the first time by the author, a reduced risk in running successful simulations is appreciated and therefore RSM models are not considered for the simulations ran in this thesis.

# 6. RESULTS AND DISCUSSION

In this chapter a closer look of the results computed from the data, calculated with OpenFOAM, is done. The heat transfer and pressure drop characteristics of the two fin shapes are compared with j- and f-factor and then the flow field is analysed more carefully with streamline, glyph and contour plots. All the simulations were carried out on the Merope-cluster which is located at the Tampere Center for Scientific Computing which provides computational resources, scientific software and support for researchers at Tampere University of Technology. Calculation times with 60 cores ranged from around 5 hours to up to 100 hours depending on the cell count and inlet velocity conditions.

## 6.1 Model comparison

For this thesis, no turbulence model comparison was done because Anna Hansen from Alborg University (Hansen, 2008) has done a similar study for fin-and-tube heat exchangers with OpenFOAM where Hansen concluded that for the lower spectrum of the studied Reynolds number a dummy turbulent model called *laminar* gives the most accurate solution but at the higher velocities $k-\omega$ SST gives best results. Margarete also compared $k-\epsilon$ model but concluded that it gives the worst results with all inlet velocities. Finally she stated that overall the most accurate performance can be achieved by using $k-\omega$ SST through out the whole Reynolds spectrum and this is why this same turbulence model is chosen in this thesis for all the simulations.

### 6.1.1 Validation of the computational domain

For the validation of the computational domain used in this thesis, the results of the experiments done by Wang et al. (1996) was compared with results computed with OpenFOAM. In figure 6.1 an illustration of the computed j-factor respect to the Reynolds number with both structured and unstructured mesh is done.

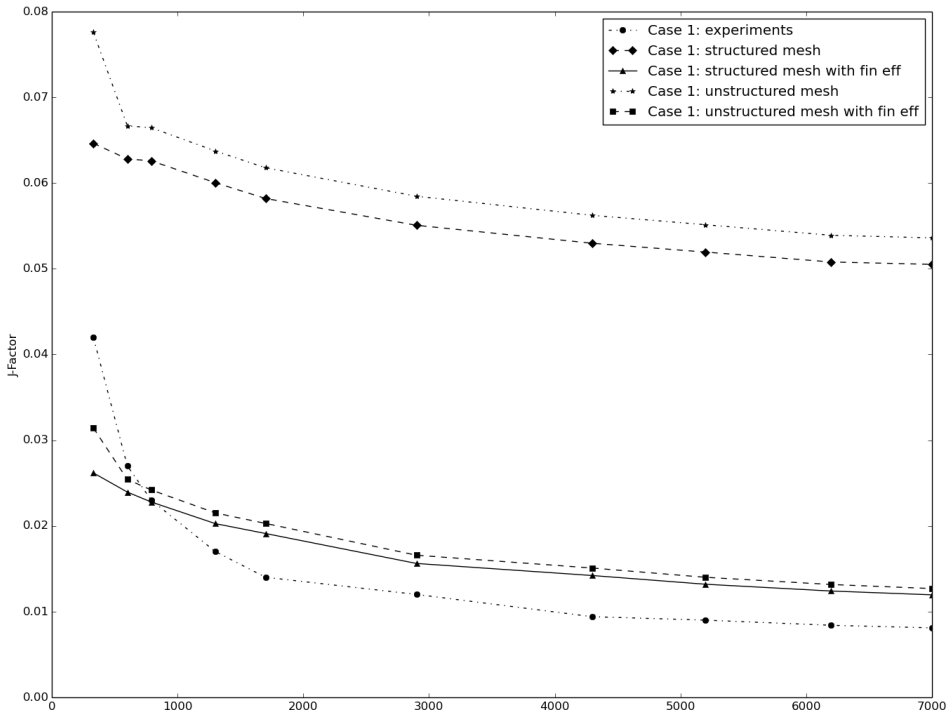



***Figure 6.1*** *Comparison of all the j-factor plots from Case 1*

Previously estimated higher trend of j-factor with the unstructured mesh can be seen on the upper half of the figure. This was expected to occur because of the influence of the entrance effect due to the fin thickness in the model of the unstructured mesh. The presence of the fin creates a boundary layer, which increases heat transfer later in the flow medium. Overall these curves posses the same trend which is essential for the validation of the unstructured mesh respect to the curvilinear structured mesh. If we include the fin efficiency introduced previously (2.2.1) we can see that the values of the j-factor correlate quite well with the experiments. The simulated results seem to have a higher value everywhere else but not with low Reynolds numbers. This could be because of the low value of the modelled turbulence with low flow velocities. Another efficiency parameter that was calculated is the fanning friction factor. In figure 6.2 is an illustration of the results from both of the mesh types.

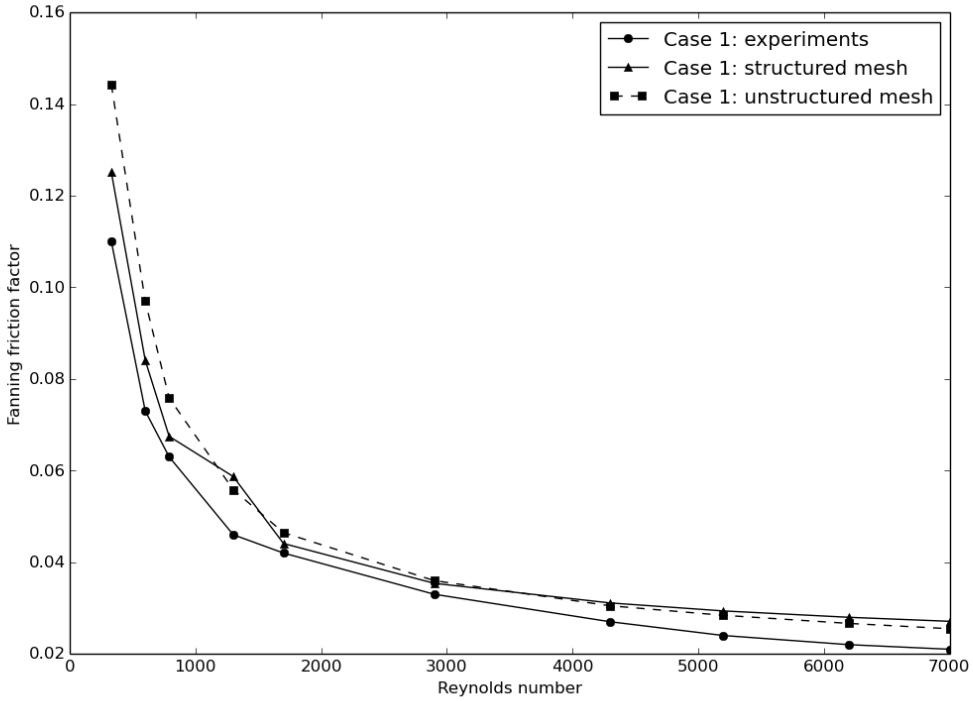



***Figure 6.2*** *Comparison of all the fanning friction factor plots from case 1*

Structured mesh seems to correlate closer with the experiments than the unstructured mesh. But this is not the case with higher flow velocities where the unstructured mesh performs better. It was seen that much more calculation points would be needed for a more precise comparison.

## 6.2 Comparison of the efficiencies of the two fin shapes

If we now compare the efficiency parameters of the two studied fin shapes and tube configurations, some very interesting characteristics can be found. If we look at figure 6.3, where the j-factor is plotted without any fin efficiencies, the plain fin shape seems to transfer heat more efficiently than the x-slit fin.

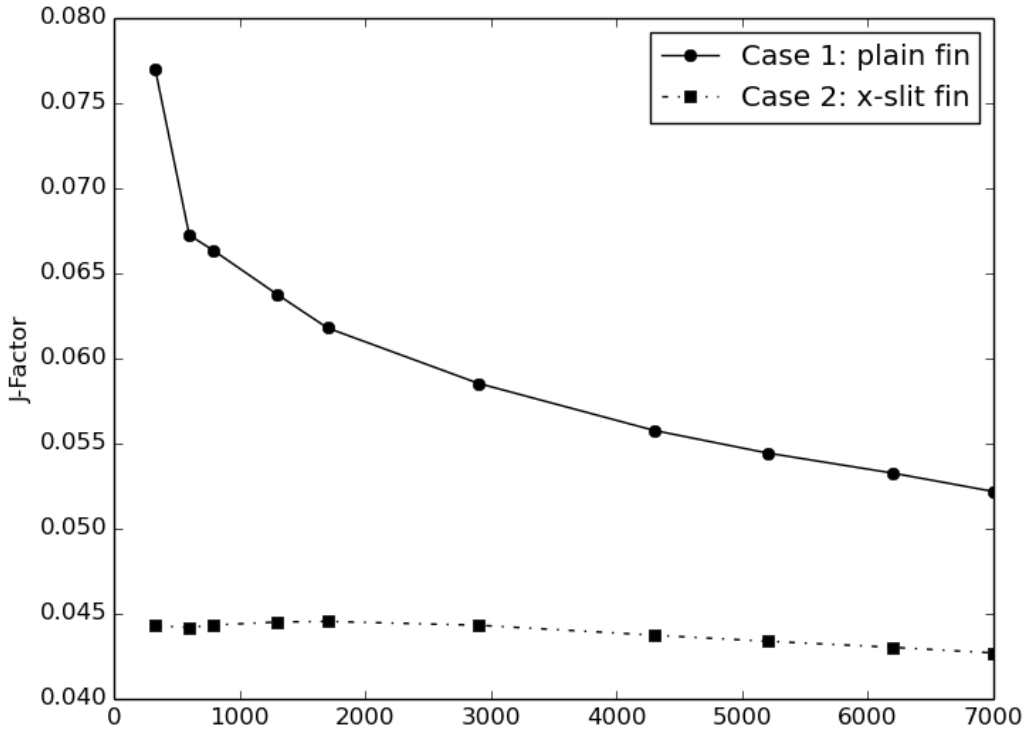



***Figure 6.3*** *Comparison of the j-factor between plain and x-slit fin*

This is due to the more dense disposition and the size of the tubes. It must be remembered that even though the temperature at the outlet rises higher with the x-slit fin shape(as can be seen in figure 6.9), compared to the plain fin geometry, a much higher potential mass thermal capacity flows through it due to the bigger frontal face area. If other parameters would have been kept constant and only the fin shape would have been changed, it would be sure that x-slit fin shape would transfer more heat than the plain fin geometry. If we now look at the normalised pressure drop in figure 6.4 we see that x-slit fin has a much higher fanning friction factor with all the flow velocities studied in this thesis.

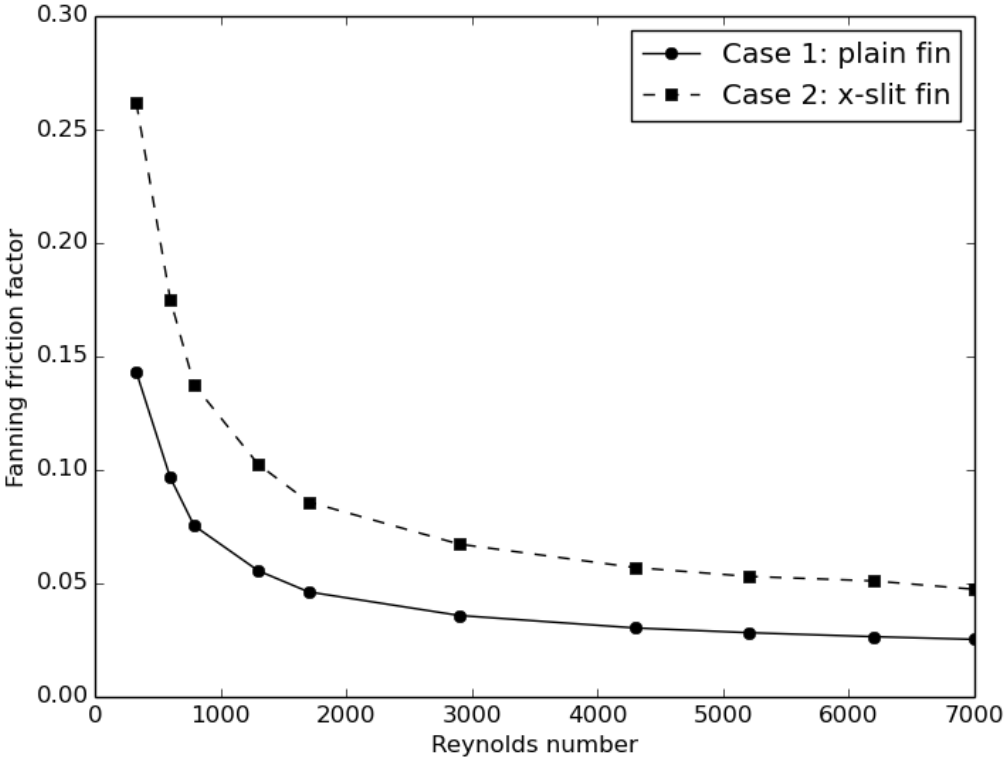



***Figure 6.4*** *Comparison of the fanning friction factor between plain and x-slit fin*

This means that proportionally the more dense disposition of the tubes does not create as high pressure drop as the x-slit fin creates with the wider geometry. Of course this kind of comparison makes little sense in normal engineer applications but it is made here just to emphasize the importance of the efficiency parameters and their importance in comparing different fin shapes and their overall heat transfer enhancement capabilities.

## 6.3 Flow characteristics

If we look closer at the flow structures between the fins and before the first tube we can say that it looks to be combined from two fundamental fluid study cases: channel flow and an impingement flow. The features of both of these can be seen on figure 6.5.

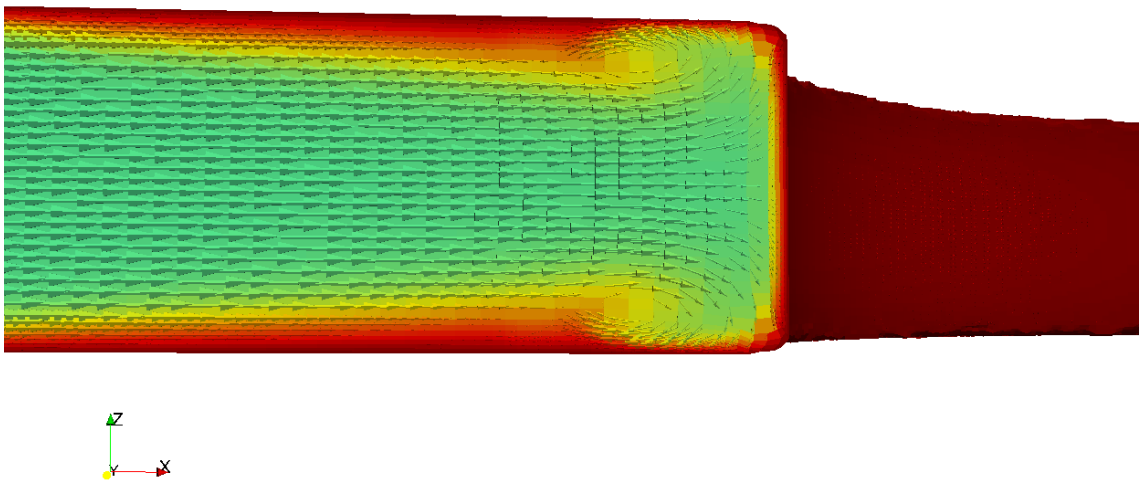



***Figure 6.5*** *Impignment flow on the first tube*

On the left side of the picture the velocity profile can be seen to have its curvy shape as the velocity reaches its maximum value on the middle of the channel and gradually goes to zero on the wall. On the right side of the picture before the round shape of the tube, an impingement flow and its two vortices can be seen on both upper and lower parts of the channel. Recognizing these kind of basic flow features can be highly useful in the process of validating the computational domain but also more importantly when thinking about enhancing the heat transfer and utilizing all the research done on these subjects.

### 6.3.1 Comparing flow structures with glyphs, streamlines and Q-criteria contour plots

If we want to look more closely of the flow structures between the fins some post-processing of the calculated data needs to be done. For this an open source program called Paraview and first the tool called Glyph was used. In figures 6.6 and 6.7 are

the glyph fields coloured with velocity and temperature in the plain that was cut from the middle of the channel, respectively.

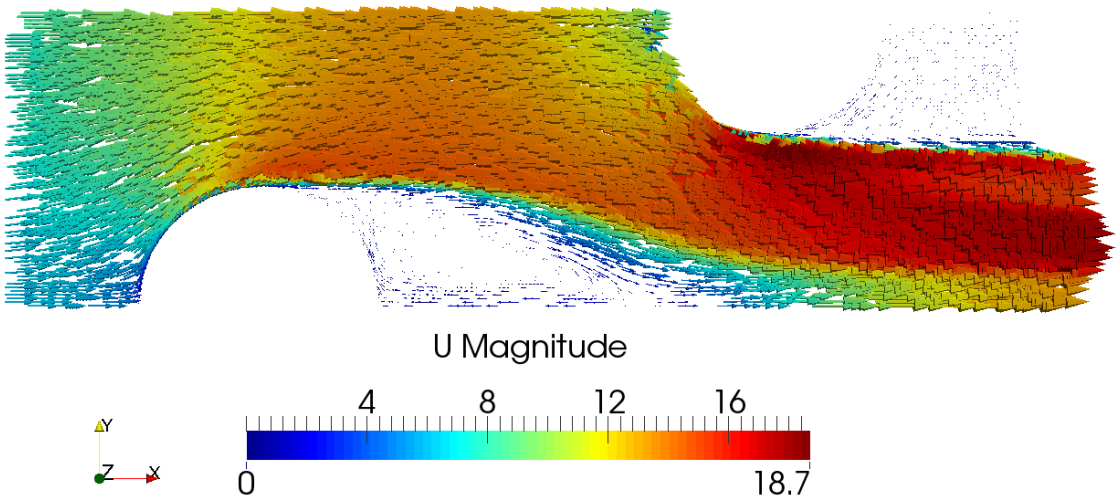



***Figure 6.6*** *Vector plot coloured with velocity in the middle of the channel in the Case 1: plain fin*

Glyphs are vectors, whose size, shape and colour can be changed according to the purpose. Here the length of the vector is defined as the velocity of the field and the colour is set to first show the value of the velocity and then the temperature.

We can see that even though behind the tubes the temperature increases and therefore heat is transferred, the velocity in this part of the field is small compared to the mean flow field. Therefore the convective heat transfer is very low in this area.

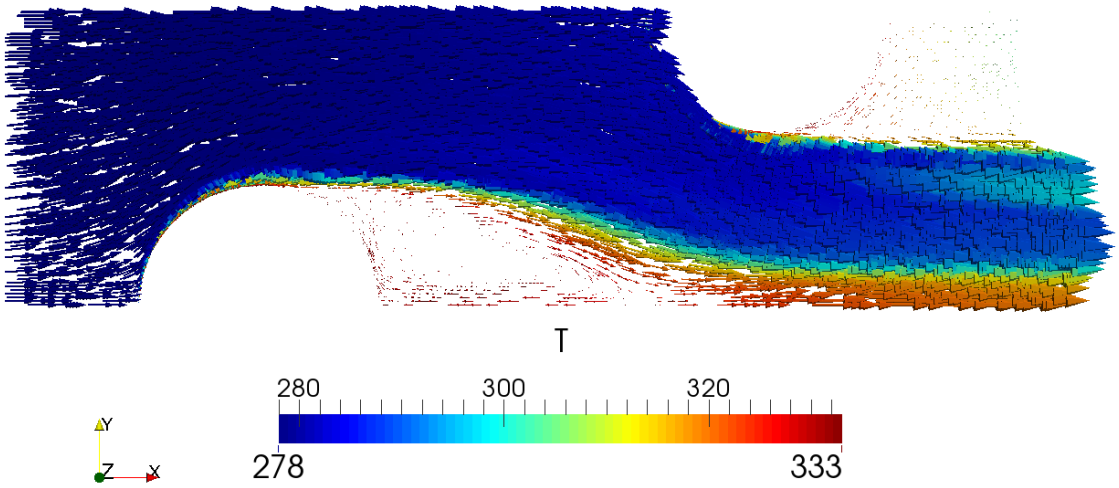



***Figure 6.7*** *Vector plot coloured with temperature in the middle of the channel in the Case 1: plain fin*

It can be also concluded that not much vorticity can be seen in the mean flow field where the main part of the air is flowing. This means that the mixing of the flow is low and the boundary layers on both sides of the channel are not merged in the middle. If the fin spacing would be decreased at some point the boundary layers would blend in to each other. This would increase the heat transfer but at the same time drastically increase the pressure drop in the air.

If we then compare the situation to the one in the x-slit fin glyph plots in figures 6.8 and 6.9 from the same plane in the middle of the channel, we can see that first of all the recirculation zone is much smaller compared to the plain fin.

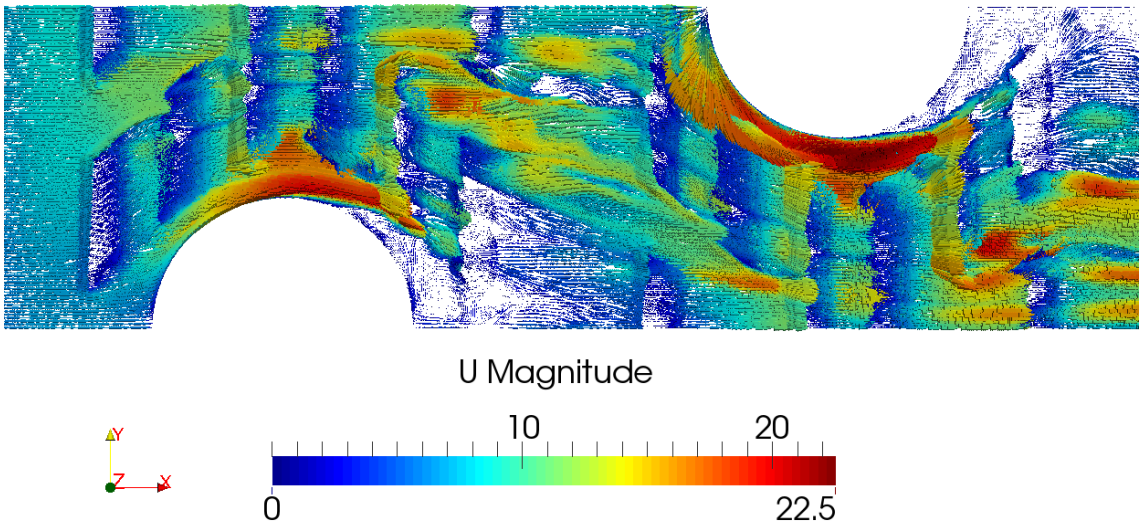



***Figure 6.8*** *Vector plot coloured with velocity in the middle of the channel in the Case 2: x-slit fin*

And second, the velocity distribution is overall more even throughout the channel. This facilitates the convective heat transfer but when the flow is distracted of its path, momentum energy of the air is lost and pressure drop is increased.

If we think about the boundary layers in the flow field. It can be seen that the boundary layers keep reforming again and again while the air flow encounters the slits in the fin. This is known to increase the heat transfer and the peak of the heat transfer rate is known to be located at the beginning at the boundary layer.

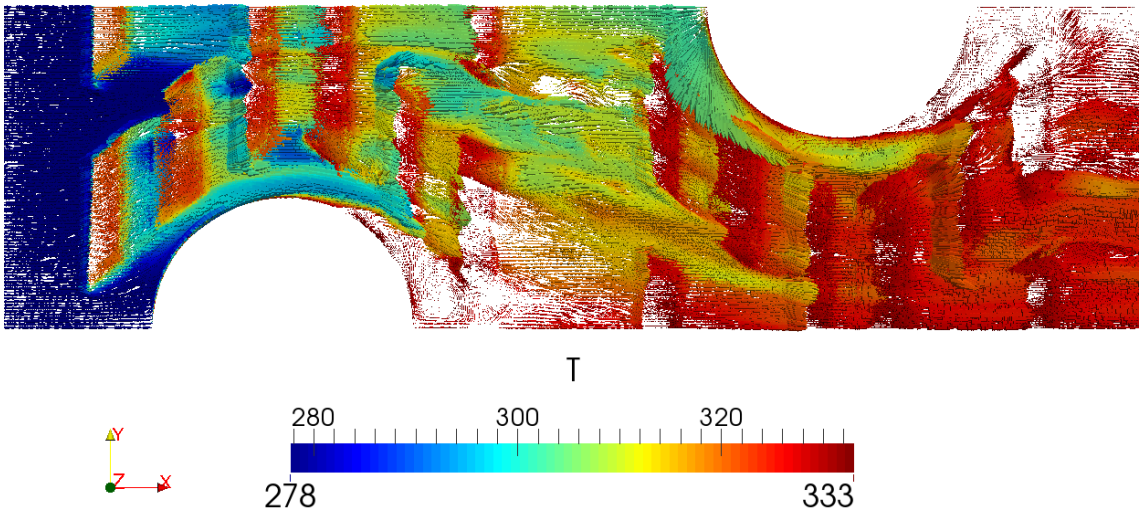



***Figure 6.9*** *Vector plot coloured with temperature in the middle of the channel in the Case 2: x-slit fin*

If we compare figure 6.7 with figure 6.9 and the temperature distribution after the fin. It can be clearly seen that the temperature of the air increases much higher with the x-slit fin than with the plain fin. The mass averaged temperature after the fin region with the x-slit fin is around 326K and for plain fin 302K with the Reynolds number value of 7000. At the same time the pressure drop increases from 120Pa with the plain fin to 366Pa with the x-slit fin.

One can illustrate the mixing of the air with for example plotting the steady-state streamlines in the flow field as in figure 6.10.

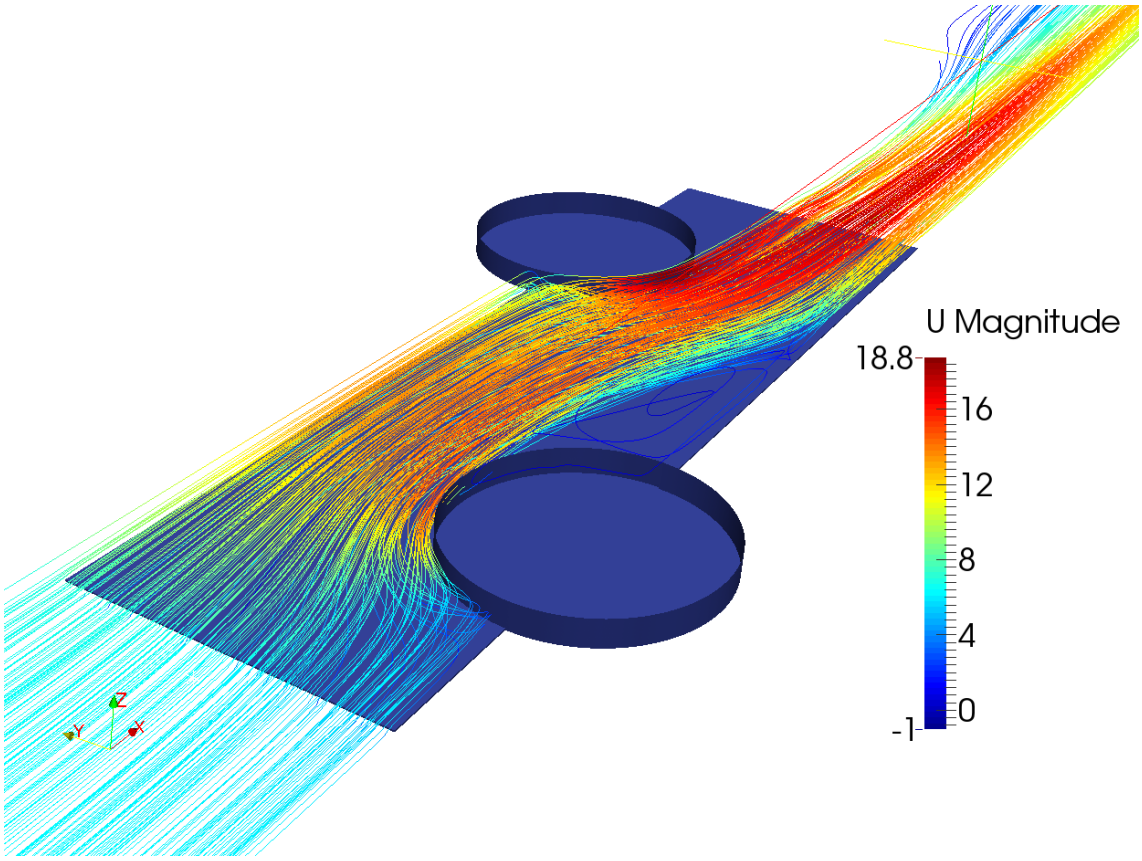



***Figure 6.10*** *Streamline-plot of the velocity field in Case 1: plain fin*

We can see that almost no vortices can be found in the flow field, except in the presence of the tubes. Neither no sign of the entrance effect can be seen in the picture.

Of course a lot of the vortex structures are smoothed out because of the steady-state nature of the solver rhoSimpleFoam, which does not resolve the time dependent fluctuations in the flow field. The turbulent model $k-\omega$ SST also smooths the flow field because a big portion of the eddies in the flow field is modelled as was explained in Chapter 5.

If we then look at the streamlines coloured with velocity in the x-slit fin shape, we can see that the raised slits create vortices every time the flow passes through them.

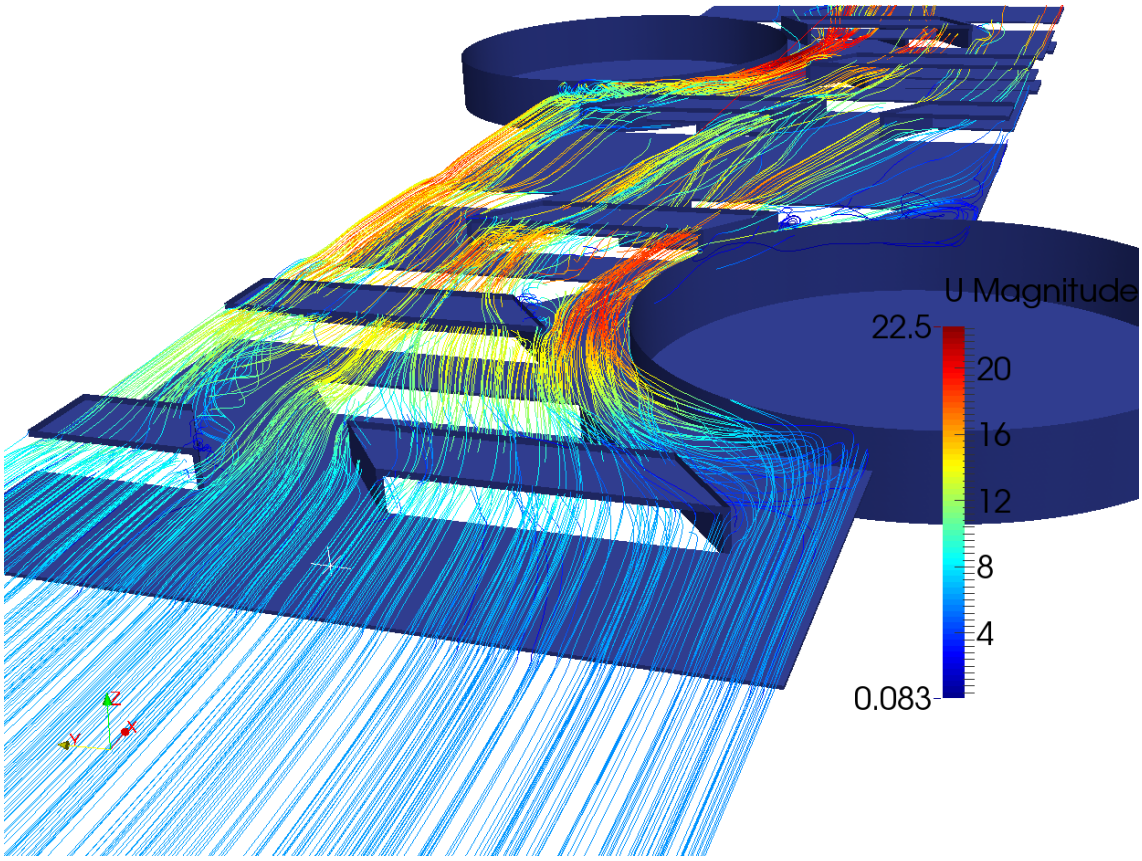



***Figure 6.11*** *Streamline-plot of the velocity field in Case 2: x-slit fin*

It should be noted that the figures are plotted only of the lower half of the channel for clearer illustration. The vortices can be seen to form behind the slits and the flow is guided through the channel evenly so that the heat transfer is kept on a higher level on the whole channel compared to the plain fin.

On the left side of figure 6.11, behind the first slit, a longitudal vortex structure can be seen. This structure was encountered behind all the slit corners in the x-slit fin geometry. As discussed before, this is the same vortex structure that was encountered with the vortex generators and it is known to be a very good way to enhance heat transfer in fin-and-tube heat exchangers.

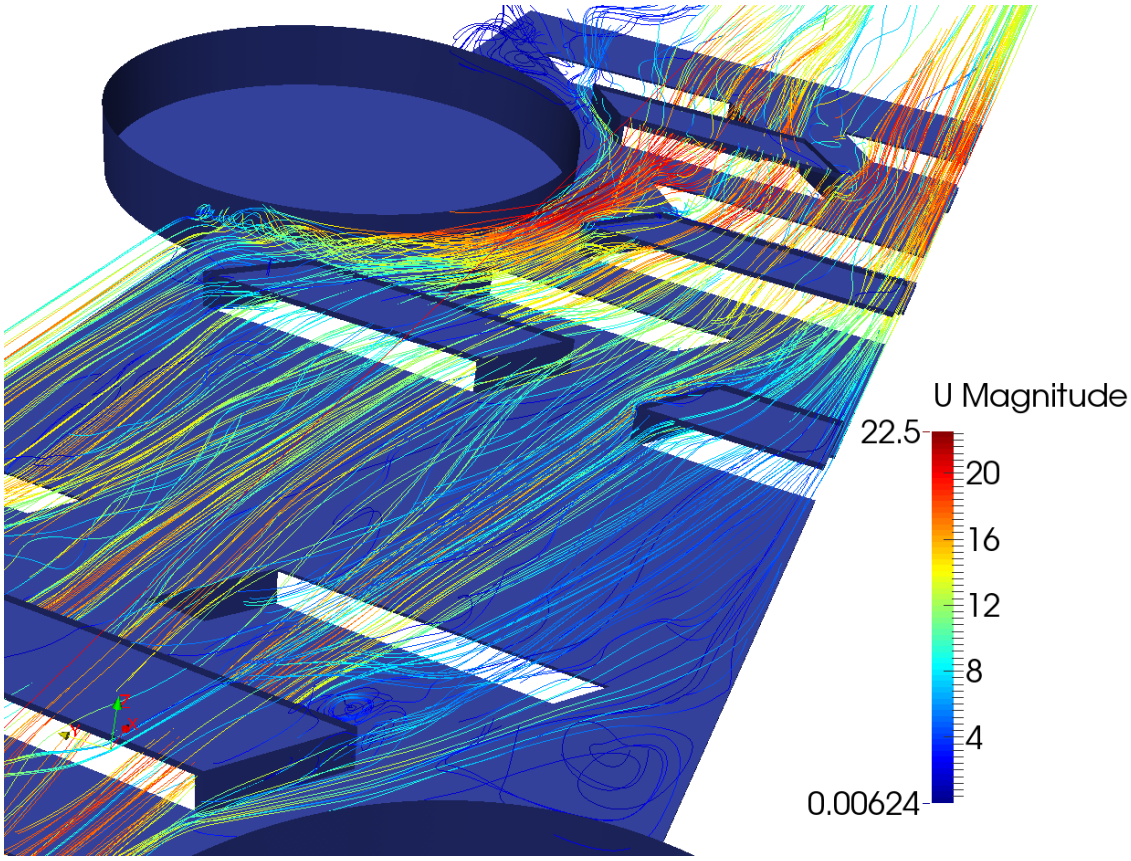



***Figure 6.12*** *Streamline-plot of the velocity field in Case 2: x-slit fin*

The vortices behind the slits can be seen more closely here in figure 6.12. This gives a better heat transfer enhancement than a sine-wave or a herringbone fin where all the vortices are transverse-kind. If we do not pay attention now to the direction of the axis in the vortices we can compare these two types of fin shapes with another way to illustrate the vortices. This could be for example the Q-criterion (Chakraborty et al., 2005), which is defined as the second invariant of the velocity gradient tensor (Hunt et al., 1988) that can be calculated from the flow field with a utility pre-installed in OpenFOAM. Which represents vorticity in a point and a good way to illustrate this value is to make a contour plot of it. An example of this can be seen in figures 6.13 and 6.14, for the plain and x-slit fin, respectively.

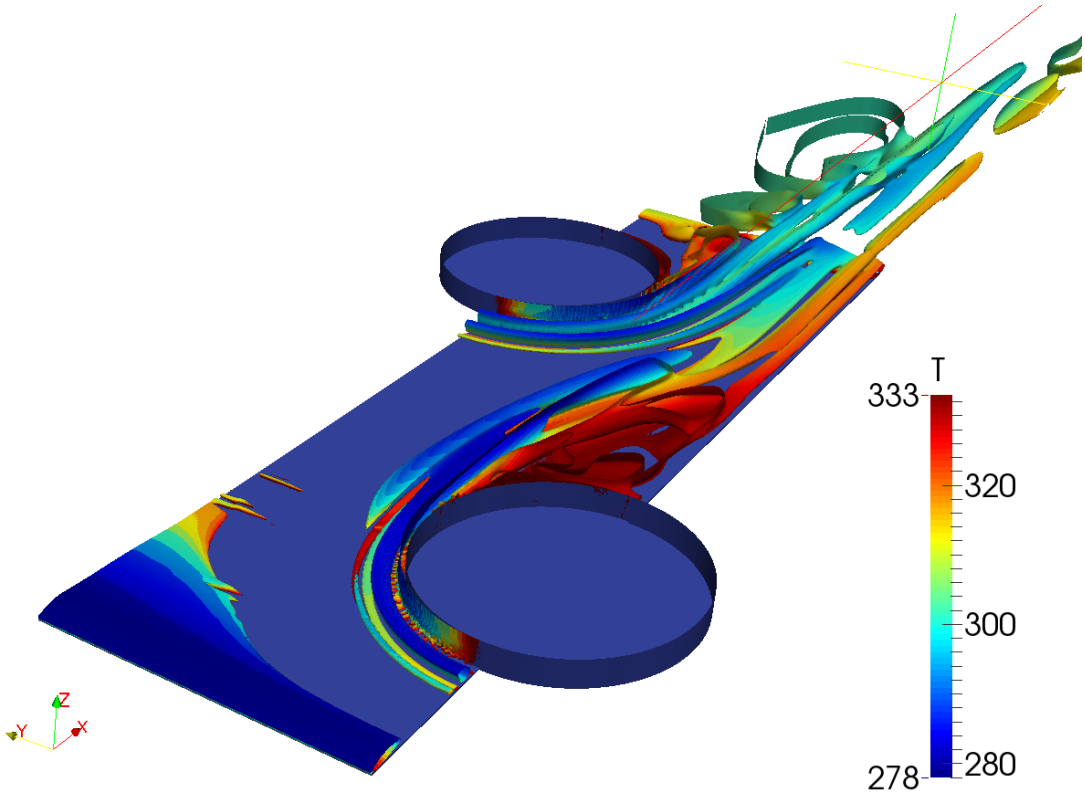



***Figure 6.13*** *Q-criteria contour plot of the Case 1: plain fin*

It can be said that the plain fin shape has a small vortex on the boundary layer at the entrance of the fin region, which can be seen on the lower left corner of the figure 6.13. Another set of vortices can be seen to go around the tubes. These are the same vortices that were found earlier in section 6.3. Some more vortex structures can be found behind the tubes but on the fin region, next to the first tube and overall upstream from the second tube, no vortices exists. If we then look at the Q-criteria contour plot of the x-slit fin shape in figure 6.14

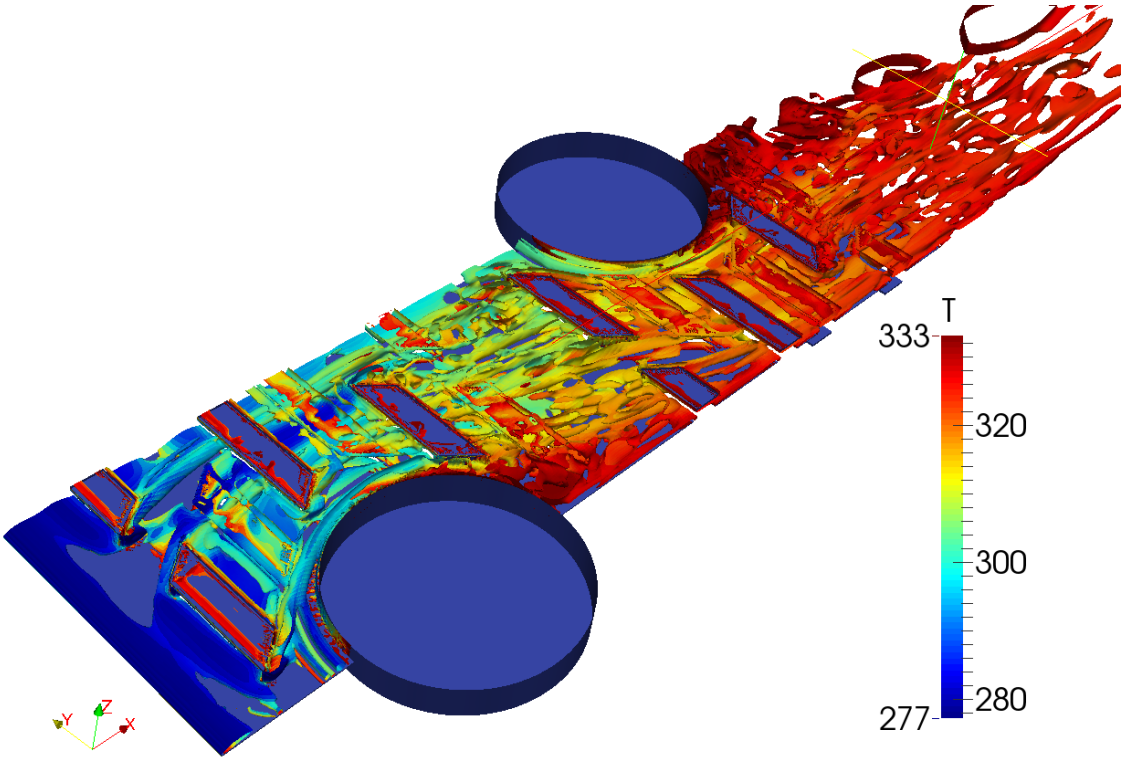



***Figure 6.14*** *Q-criteria contour plot of the Case 2: x-slit fin*

We can see that vortices are forming throughout the flow domain. As said before this is the main reason for higher heat exchange as well as pressure drop in the flow domain than a plain fin of the same size would have.

It should be noted here that since we use a RANS-model to model the turbulent structures and the solver is a steady-state solver, these Q-criterion plots does not represent any physical flow structure but are just a sign of overall level of turbulence in the flow.

# 7. CONCLUSION

In this Master's thesis project a simulation method for convective heat transfer on the air side of a fin-and-tube heat exchanger was developed. All the programs used were open source and distributed with the GNU public license. The method developed is fully capable of simulating arbitrary fin shapes and comparing the hydrothermal characteristics of different fin shapes. Two different fin shapes: plain and x-slit fin with differing tube configurations were studied. Plain fin case was used as a model validation case and two different types of meshes were created for this fin shape: a structured curvilinear mesh and an unstructured mesh. This unstructured mesh creation method was developed because of the increased complexity on the fin shape and its influence to the difficulty level in the meshing procedure with structured hand made meshes. A lot of problems were encountered with the meshing procedure for the unstructured mesh, as snappyHexMesh is prone to crashing for many different reasons. These reasons are accumulated over the different meshing phases in the meshing procedure of the snappyHexMesh. Main reasons are too coarse background mesh used in the beginning and the poor quality of the snapping process if the parameters are set poorly.

It was found that it is possible and fairly efficient to study fin-and-tube heat exchanger fin shapes with open source software, once the selection of turbulence model, solvers and boundary conditions are clear. The most difficult one to make work properly is the boundary condition called cyclicAMI which needs special attention with its parameter settings. None of the open source meshing tools studied in this thesis can compete with the commercial ones on its own, but it was seen that all of them are good for some specific meshing task. So it could be said that combining all the open source meshing tools and all their features available now in the summer of 2015 together, they offer a fairly good and free alternative.

It was shown that both structured and unstructured meshes correlate quite well with the experiments. Even the heat transfer after the fin efficiency is included in to the calculations. More deviation was seen on the heat transfer comparison but

the pressure drop was seen to correlate with a really good accuracy. This means that it can be concluded that comparing different fin shapes of a fin-and-tube heat exchanger with open source software is possible.

Then a plain fin geometry with smaller tube size was compared to x-slit fin geometry with a bigger tube size and disposition of the tubes. It was noticed that the heat transfer efficiency parameter j-factor, with ten different flow velocities, was higher for the plain fin than for the x-slit fin even though x-slit fin opposes a lot more turbulent structures in the flow field. Then the situation was completely turned around when the normalised pressure drop called fanning friction factor parameter was compared with ten different inlet velocities. It was found to be higher for the x-slit fin than for the plain fin. This is due to the non-disturbed flow field in the plain fin, even though the tubes are smaller and closer to each other. It is clear that the efficiency parameters are an important tool in the process of comparing different fin shapes and that comparing only one output variable like temperature should always be avoided.

The next phase that should be done in this case of study is to eliminate the uncertainties caused by the fin efficiency approximation and include the conduction heat transfer in the fins and the coupled heat transfer from the fins to the air. This will bring the simulation closer to the reality and give more accurate results for all kinds of fin geometries simulated with the simulation method developed in this thesis project.

## References

Gnu general public license, 2015. URL `http://www.gnu.org/licenses/gpl.html`.

P. Bradshaw, T. Cebeci, and J. H. Whitelaw. Engineering calculation methods for turbulent flow. *NASA STI/Recon Technical Report A*, 82:20300, 1981.

P. Chakraborty, S. Balachandar, and R. J. Adrian. On the relationships between local vortex identification schemes. *Journal of Fluid Mechanics*, 535:189–214, 2005.

R. H. Chiang and J. L. Esformes. Heat exchanger tube, July 26 1994. US Patent 5,332,034.

B.-N. Choi, F. Yi, H.-M. Sim, and N.-H. Kim. Air-side performance of fin-and-tube heat exchangers having sine wave fins and oval tubes. *Korean Journal of Air-Conditioning and Refrigeration Engineering*, 25(5):279–288, 2013.

L. Davidson. *Fluid mechanics, turbulent flow and turbulence modeling.* Division of Fluid Dynamics, Chalmers University of Technology, 2015.

A. Erek, B. Özerdem, L. Bilir, and Z. Ilken. Effect of geometrical parameters on heat transfer and pressure drop characteristics of plate fin and tube heat exchangers. *Applied Thermal Engineering*, 25(14):2421–2431, 2005.

E. Fornasieri and L. Mattarolo. Air-side heat transfer and pressure loss in finned tube heat exchangers: state of art. *Proceedings of the European Conference on Finned Tube Heat Exchangers*, 1991.

W. Frank M. *Fluid Mechanics.* McGraW-Hill, 5 edition, 2003. ISBN 0-07-121566-2.

C. Geankoplis. *Transport processes and separation process principles (includes unit operations).* Prentice Hall Press, 2003.

L. Goldstein and E. Sparrow. Experiments on the transfer characteristics of a corrugated fin and tube heat exchanger configuration. *Journal of Heat Transfer*, 98(1): 26–34, 1976.

A. M. Hansen. Cfd simulation of a fin-and-tube heat exchanger. Master's thesis, Aalborg Universitet, 2008.

J. C. R. Hunt, A. Wray, and P. Moin. Eddies, stream, and convergence zones in turbulent flows. *Proceeding of the Summer Program 1988*, 1988.

J.-Y. Jang, M.-C. Wu, and W.-J. Chang. Numerical and experimental studies of threedimensional plate-fin and tube heat exchangers. *International Journal of Heat and Mass Transfer*, 39(14):3057–3066, 1996.

J. Jeong, C. N. Kim, and B. Youn. A study on the thermal contact conductance in fin–tube heat exchangers with 7mm tube. *International journal of heat and mass transfer*, 49(7):1547–1555, 2006.

A. Joardar and A. Jacobi. Heat transfer enhancement by winglet-type vortex gen-

erator arrays in compact plain-fin-and-tube heat exchangers. *International Journal of refrigeration*, 31(1):87–97, 2008.

B. Launder and B. Sharma. Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in heat and mass transfer*, 1(2):131–137, 1974.

J.-S. Leu, Y.-H. Wu, and J.-Y. Jang. Heat transfer and fluid flow analysis in plate-fin and tube heat exchangers with a pair of block shape vortex generators. *International Journal of Heat and Mass Transfer*, 47(19):4327–4338, 2004.

X.-w. Li, H. Yan, J.-a. Meng, and Z.-x. Li. Visualization of longitudinal vortex flow in an enhanced heat transfer tube. *Experimental thermal and fluid science*, 31(6): 601–608, 2007.

C.-W. Lu, J.-M. Huang, W. Nien, and C.-C. Wang. A numerical investigation of the geometric effects on the performance of plate finned-tube heat exchanger. *Energy Conversion and Management*, 52(3):1638–1643, 2011.

A. Mills. *Basic Heat and Mass Transfer*. Prentice Hall, 1999. ISBN 9780130962478. URL `https://books.google.fi/books?id=C5AvAQAAIAAJ`.

M. S. Mon and U. Gross. Numerical study of fin-spacing effects in annular-finned tube heat exchangers. *International journal of heat and mass transfer*, 47(8):1953–1964, 2004.

S. B. Pope. *Turbulent flows*. Cambridge university press, 2000.

O. Reynolds. An experimental inves- tigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. 1883.

R. Romero-Méndez, M. Sen, K. Yang, and R. McClain. Effect of fin spacing on convection in a plate fin and tube heat exchanger. *International Journal of Heat and Mass Transfer*, 43(1):39–51, 2000.

K. Ryu, S.-J. Yook, and K.-S. Lee. Optimal design of a corrugated louvered fin. *Applied Thermal Engineering*, 68(1-2):76 – 79, 2014. ISSN 1359-4311. doi: http://dx.doi.org/10.1016/j.applthermaleng.2014.04.022. URL `http://www.sciencedirect.com/science/article/pii/S1359431114002865`.

T. Schmidt. Heat transfer calculations for extended surfaces. *Heat Transfer 1978*, 4(0):193 – 199, 1949. URL `http://www.scopus.com/record/display.url?eid=2-s2.0-0010728386&origin=inward&txGid=E1CA149D625099D370BA2D4575A50C0B.FZg2ODcJC9ArCe8WOZPvA%3a6#`.

D. Tang, Y. Peng, and D. Li. Numerical and experimental study on expansion forming of inner grooved tube. *Journal of Materials Processing Technology*, 209

(10):4668–4674, 2009.

H. Tennekes and J. L. Lumley. *A first course in turbulence*. MIT press, 1972.

M. Tutar and A. Akkoca. Numerical analysis of fluid flow and heat transfer characteristics in three-dimensional plate fin-and-tube heat exchangers. *Numerical Heat Transfer, Part A: Applications*, 46(3):301–321, 2004.

B. Venkanna. *Fundamentals of heat and mass transfer*. PHI Learning Pvt. Ltd., 2010.

H. K. Versteeg and W. Malalasekera. *An Introduction to Computational Fluid Dynamics*. McGraW-Hill, 2 edition, 2007. ISBN 978-0131274983.

C.-C. Wang, Y.-J. Chang, Y.-C. Hsieh, and Y.-T. Lin. Sensible heat and friction characteristics of plate fin-and-tube heat exchangers having plane fins. *International Journal of Refrigeration*, 19(4):223 – 230, 1996. ISSN 0140-7007. doi: http://dx.doi.org/10.1016/0140-7007(96)00021-7. URL `http://www.sciencedirect.com/science/article/pii/0140700796000217`.

C.-C. Wang, K.-Y. Chi, Y.-J. Chang, and Y.-P. Chang. An experimental study of heat transfer and friction characteristics of typical louver fin-and-tube heat exchangers. *International journal of heat and mass transfer*, 41(4):817–822, 1998.

C.-C. Wang, C.-J. Lee, C.-T. Chang, and S.-P. Lin. Heat transfer and friction correlation for compact louvered fin-and-tube heat exchangers. *International Journal of Heat and Mass Transfer*, 42(11):1945 – 1956, 1999a. ISSN 0017-9310. doi: http://dx.doi.org/10.1016/S0017-9310(98)00302-0. URL `http://www.sciencedirect.com/science/article/pii/S0017931098003020`.

C.-C. Wang, W.-H. Tao, and C.-J. Chang. An investigation of the airside performance of the slit fin-and-tube heat exchangers. *International Journal of Refrigeration*, 22(8):595 – 603, 1999b. ISSN 0140-7007. doi: http://dx.doi.org/10.1016/S0140-7007(99)00031-6. URL `http://www.sciencedirect.com/science/article/pii/S0140700799000316`.

C.-C. Wang, W.-H. Tao, and C.-J. Chang. An investigation of the airside performance of the slit fin-and-tube heat exchangers. *International Journal of Refrigeration*, 22(8):595–603, 1999c.

C.-C. Wang, K.-Y. Chi, and C.-J. Chang. Heat transfer and friction characteristics of plain fin-and-tube heat exchangers, part ii: Correlation. *International Journal of Heat and Mass Transfer*, 43(15):2693–2700, 2000a.

C.-C. Wang, R. L. Webb, and K.-Y. Chi. Data reduction for air-side performance of fin-and-tube heat exchangers. *Experimental Thermal and Fluid Science*, 21(4): 218–226, 2000b.

C.-C. Wang, W.-S. Lee, and W.-J. Sheu. A comparative study of compact enhanced fin-and-tube heat exchangers. *International Journal of Heat and Mass Transfer*, 44 (18):3565–3573, 2001.

C.-C. Wang, Y.-M. Hwang, and Y.-T. Lin. Empirical correlations for heat transfer and flow friction characteristics of herringbone wavy fin-and-tube heat exchangers. *International Journal of Refrigeration*, 25(5):673–680, 2002a.

C.-C. Wang, J. Lo, Y.-T. Lin, and C.-S. Wei. Flow visualization of annular and delta winlet vortex generators in fin-and-tube heat exchanger application. *International Journal of Heat and Mass Transfer*, 45(18):3803–3815, 2002b.

R. Webb. *Principles of Enhanced Heat Transfer*. Number nid. 1 in Principles of Enhanced Heat Transfer. Taylor & Francis, 2005. ISBN 9781591690146. URL `https://books.google.fi/books?id=Yn1gQgAACAAJ`.

S. Wongwises and Y. Chokeman. Effect of fin pitch and number of tube rows on the air side performance of herringbone wavy fin and tube heat exchangers. *Energy Conversion and Management*, 46(13 - 14):2216 – 2231, 2005. ISSN 0196-8904. doi: http://dx.doi.org/10.1016/j.enconman.2004.09.011. URL `http://www.sciencedirect.com/science/article/pii/S0196890404002584`.

X. Wu, W. Zhang, Q. Gou, Z. Luo, and Y. Lu. Numerical simulation of heat transfer and fluid flow characteristics of composite fin. *International Journal of Heat and Mass Transfer*, 75(10):414–424, 2014. doi: http://dx.doi.org/10.1016/j.ijheatmasstransfer.2014.03.087.

G. Xie, Q. Wang, and B. Sunden. Parametric study and multiple correlations on air-side heat transfer and friction characteristics of fin-and-tube heat exchangers with large number of large-diameter tube rows. *Applied Thermal Engineering*, 29(1):1 – 16, 2009. ISSN 1359-4311. doi: http://dx.doi.org/10.1016/j.applthermaleng.2008.01.014. URL `http://www.sciencedirect.com/science/article/pii/S135943110800029X`.

V. Yakhot, S. Orszag, S. Thangam, T. Gatski, and C. Speziale. Development of turbulence models for shear flows by a double expansion technique. *Physics of Fluids A: Fluid Dynamics (1989-1993)*, 4(7):1510–1520, 1992.

# APPENDIX A. THE PYTHON SCRIPT THAT IS USED FOR GEOMETRY CREATION OF THE MODEL VALIDATION CASE.

```
#Model validation geometry generation
#Import all the required interfaces
import salome
salome.salome_init()
import GEOM
from salome.geom import geomBuilder
geompy = geomBuilder.New(salome.myStudy)
import math

#Create the axels
OZ     = geompy.MakeVectorDXDYDZ(0,0,1)
OX     = geompy.MakeVectorDXDYDZ(1,0,0)
OY     = geompy.MakeVectorDXDYDZ(0,1,0)
geompy.addToStudy(OZ, "OZ")
geompy.addToStudy(OY, "OY")
geompy.addToStudy(OX, "OX")

#Create the parametric dimensions
H=0.00224
Pl=0.022
Pt=0.0254

#Centerpoint for the first circle
R= 0.005115
cp1X = Pl/2
cp1Y = 0

#Centerpoint for the second circle
cp2X = 1.5*Pl
cp2Y = Pt*0.5
```

```
#Inner circle radius divided into two
X= R/math.sqrt(2)
Y= R/math.sqrt(2)

#Outer circle radius divided into two
R2= R*1.2
dX = R2/math.sqrt(2)
dY = R2/math.sqrt(2)

#Vertices for geometry building

#Centerpoint for circle 1
pnt1 = geompy.MakeVertex (cp1X,cp1Y,0)
#Inner circle1
pnt2 = geompy.MakeVertex (cp1X-R,cp1Y,0)
pnt3 = geompy.MakeVertex (cp1X-X,cp1Y+Y,0)
pnt4 = geompy.MakeVertex (cp1X+X,cp1Y+Y,0)
pnt5 = geompy.MakeVertex (cp1X+R,cp1Y,0)
#outer circle1
pnt6 = geompy.MakeVertex (cp1X-R2,cp1Y,0)
pnt7 = geompy.MakeVertex (cp1X-dX,cp1Y+dY,0)
pnt8 = geompy.MakeVertex (cp1X+dX,cp1Y+dY,0)
pnt9 = geompy.MakeVertex (cp1X+R2,cp1Y,0)

#centerpoint for circle 2
pnt10 = geompy.MakeVertex (cp2X,cp2Y,0)
#Inner circle2
pnt11 = geompy.MakeVertex (cp2X-R,cp2Y,0)
pnt12 = geompy.MakeVertex (cp2X-X,cp2Y-Y,0)
pnt13 = geompy.MakeVertex (cp2X+X,cp2Y-Y,0)
pnt14 = geompy.MakeVertex (cp2X+R,cp2Y,0)
#outer circle2
pnt15 = geompy.MakeVertex (cp2X-R2,cp2Y,0)
pnt16 = geompy.MakeVertex (cp2X-dX,cp2Y-dY,0)
pnt17 = geompy.MakeVertex (cp2X+dX,cp2Y-dY,0)
pnt18 = geompy.MakeVertex (cp2X+R2,cp2Y,0)
```

```
#cube
pnt19 = geompy.MakeVertex (0,0,0)
pnt20 = geompy.MakeVertex (0,Pt*0.5,0)
pnt21 = geompy.MakeVertex (2*Pl,Pt*0.5,0)
pnt22 = geompy.MakeVertex (2*Pl,0,0)
#Box1
pnt23 = geompy.MakeVertex (0, Pt*0.25,0)
pnt24 = geompy.MakeVertex (Pl, Pt*0.25,0)
pnt25 = geompy.MakeVertex (Pl, 0,0)

#Box2
pnt26 = geompy.MakeVertex (2*Pl, Pt*0.25,0)
pnt27 = geompy.MakeVertex (Pl, Pt*0.5,0)

#Extra points for extra arcs
pnt28 = geompy.MakeVertex (cp1X,Pt*0.35,0)
pnt29 = geompy.MakeVertex (cp2X,Pt*0.15,0)

#Extra points for freestream inlet and outlet
pnt30 = geompy.MakeVertex (-Pl,0,0)
pnt31 = geompy.MakeVertex (-Pl,0.5*Pt,0)
pnt32 = geompy.MakeVertex (7*Pl,0,0)
pnt33 = geompy.MakeVertex (7*Pl,0.5*Pt,0)

#Add points to study
geompy.addToStudy(pnt1, "pnt1")
geompy.addToStudy(pnt2, "pnt2")
geompy.addToStudy(pnt3, "pnt3")
geompy.addToStudy(pnt4, "pnt4")
geompy.addToStudy(pnt5, "pnt5")
geompy.addToStudy(pnt6, "pnt6")
geompy.addToStudy(pnt7, "pnt7")
geompy.addToStudy(pnt8, "pnt8")
geompy.addToStudy(pnt9, "pnt9")
geompy.addToStudy(pnt10, "pnt10")
```

```
geompy.addToStudy(pnt11, "pnt11")
geompy.addToStudy(pnt12, "pnt12")
geompy.addToStudy(pnt13, "pnt13")
geompy.addToStudy(pnt14, "pnt14")
geompy.addToStudy(pnt15, "pnt15")
geompy.addToStudy(pnt16, "pnt16")
geompy.addToStudy(pnt17, "pnt17")
geompy.addToStudy(pnt18, "pnt18")
geompy.addToStudy(pnt19, "pnt19")
geompy.addToStudy(pnt20, "pnt20")
geompy.addToStudy(pnt21, "pnt21")
geompy.addToStudy(pnt22, "pnt22")
geompy.addToStudy(pnt23, "pnt23")
geompy.addToStudy(pnt24, "pnt24")
geompy.addToStudy(pnt25, "pnt25")
geompy.addToStudy(pnt26, "pnt26")
geompy.addToStudy(pnt27, "pnt27")
geompy.addToStudy(pnt28, "pnt28")
geompy.addToStudy(pnt29, "pnt29")
geompy.addToStudy(pnt30, "pnt30")
geompy.addToStudy(pnt31, "pnt31")
geompy.addToStudy(pnt32, "pnt32")
geompy.addToStudy(pnt33, "pnt33")

#Lines that are created between the points
#Tube1 Connecting lines between arcs
line1 = geompy.MakeEdge(pnt2, pnt6)
line2 = geompy.MakeEdge(pnt3, pnt7)
line3 = geompy.MakeEdge(pnt4, pnt8)
line4 = geompy.MakeEdge(pnt5, pnt9)

#Tube 1 Connecting the outer arcs

line5 = geompy.MakeEdge(pnt6, pnt19)
line6 = geompy.MakeEdge(pnt7, pnt23)
line7 = geompy.MakeEdge(pnt8, pnt24)
```

```
line8 = geompy.MakeEdge(pnt9, pnt25)

#Tube1 The box
line9 = geompy.MakeEdge(pnt19, pnt23)
line10 = geompy.MakeEdge(pnt23, pnt24)
line11 = geompy.MakeEdge(pnt24, pnt25)
line12 = geompy.MakeEdge(pnt25, pnt9)
line13 = geompy.MakeEdge(pnt6, pnt19)

#Tube2 Connecting lines between arcs
line14 = geompy.MakeEdge(pnt11, pnt15)
line15 = geompy.MakeEdge(pnt12, pnt16)
line16 = geompy.MakeEdge(pnt13, pnt17)
line17 = geompy.MakeEdge(pnt14, pnt18)

#Tube 2 Connecting the outer arcs
line18 = geompy.MakeEdge(pnt15, pnt27)
line19 = geompy.MakeEdge(pnt16, pnt24)
line20 = geompy.MakeEdge(pnt17, pnt26)

#Tube 2 The box
line21 = geompy.MakeEdge(pnt21, pnt26)
line22 = geompy.MakeEdge(pnt26, pnt24)
line23 = geompy.MakeEdge(pnt24, pnt27)
line24 = geompy.MakeEdge(pnt27, pnt15)
line25 = geompy.MakeEdge(pnt18, pnt21)

#Rest of the box
line26= geompy.MakeEdge(pnt23, pnt20)
line27= geompy.MakeEdge(pnt20, pnt27)
line28= geompy.MakeEdge(pnt26, pnt22)
line29= geompy.MakeEdge(pnt22, pnt25)

#Inlet and outlet
line30= geompy.MakeEdge(pnt19, pnt30)
line31= geompy.MakeEdge(pnt30, pnt31)
```

```
line32= geompy.MakeEdge(pnt20, pnt31)
line33= geompy.MakeEdge(pnt21, pnt33)
line34= geompy.MakeEdge(pnt33, pnt32)
line35= geompy.MakeEdge(pnt22, pnt32)

#Add lines to study
geompy.addToStudy(line1, "line1")
geompy.addToStudy(line2, "line2")
geompy.addToStudy(line3, "line3")
geompy.addToStudy(line4, "line4")
geompy.addToStudy(line5, "line5")
geompy.addToStudy(line6, "line6")
geompy.addToStudy(line7, "line7")
geompy.addToStudy(line8, "line8")
geompy.addToStudy(line9, "line9")
geompy.addToStudy(line10, "line10")
geompy.addToStudy(line11, "line11")
geompy.addToStudy(line12, "line12")
geompy.addToStudy(line13, "line13")
geompy.addToStudy(line14, "line14")
geompy.addToStudy(line15, "line15")
geompy.addToStudy(line16, "line16")
geompy.addToStudy(line17, "line17")
geompy.addToStudy(line18, "line18")
geompy.addToStudy(line19, "line19")
geompy.addToStudy(line20, "line20")
geompy.addToStudy(line21, "line21")
geompy.addToStudy(line22, "line22")
geompy.addToStudy(line23, "line23")
geompy.addToStudy(line24, "line24")
geompy.addToStudy(line25, "line25")
geompy.addToStudy(line26, "line26")
geompy.addToStudy(line27, "line27")
geompy.addToStudy(line28, "line28")
geompy.addToStudy(line29, "line29")
geompy.addToStudy(line30, "line30")
```

```
geompy.addToStudy(line31, "line31")
geompy.addToStudy(line32, "line32")
geompy.addToStudy(line33, "line33")
geompy.addToStudy(line34, "line34")
geompy.addToStudy(line35, "line35")


#Tube1 Arcs
arc1 = geompy.MakeArcCenter(pnt1, pnt2, pnt3, 0)
arc2 = geompy.MakeArcCenter(pnt1, pnt3, pnt4, 0)
arc3 = geompy.MakeArcCenter(pnt1, pnt4, pnt5, 0)
arc4 = geompy.MakeArcCenter(pnt1, pnt6, pnt7, 0)
arc5 = geompy.MakeArcCenter(pnt1, pnt7, pnt8, 0)
arc6 = geompy.MakeArcCenter(pnt1, pnt8, pnt9, 0)
arc13 = geompy.MakeArc(pnt23, pnt28, pnt24 )

#Tube2 Arcs
arc7 = geompy.MakeArcCenter(pnt10, pnt11, pnt12, 0)
arc8 = geompy.MakeArcCenter(pnt10, pnt12, pnt13, 0)
arc9 = geompy.MakeArcCenter(pnt10, pnt13, pnt14, 0)
arc10 = geompy.MakeArcCenter(pnt10, pnt15, pnt16, 0)
arc11 = geompy.MakeArcCenter(pnt10, pnt16, pnt17, 0)
arc12 = geompy.MakeArcCenter(pnt10, pnt17, pnt18, 0)
arc14 = geompy.MakeArc(pnt24, pnt29, pnt26 )

geompy.addToStudy(arc1, "arc1")
geompy.addToStudy(arc2, "arc2")
geompy.addToStudy(arc3, "arc3")
geompy.addToStudy(arc4, "arc4")
geompy.addToStudy(arc5, "arc5")
geompy.addToStudy(arc6, "arc6")
geompy.addToStudy(arc7, "arc7")
geompy.addToStudy(arc8, "arc8")
geompy.addToStudy(arc9, "arc9")
geompy.addToStudy(arc10, "arc10")
geompy.addToStudy(arc11, "arc11")
```

```
geompy.addToStudy(arc12, "arc12")
geompy.addToStudy(arc13, "arc13")
geompy.addToStudy(arc14, "arc14")

#Quadrangle faces
Qface1 = geompy.MakeQuad2Edges(line9, arc4)
Qface2 = geompy.MakeQuad2Edges(arc13, arc5)
Qface3 = geompy.MakeQuad2Edges(line11, arc6)
Qface4 = geompy.MakeQuad2Edges(line21, arc12)
Qface5 = geompy.MakeQuad2Edges(arc14, arc11)
Qface6 = geompy.MakeQuad2Edges(line23, arc10)
Qface7 = geompy.MakeQuad2Edges(line27, arc13)
Qface8 = geompy.MakeQuad2Edges(line29, arc14)
Qface9 = geompy.MakeQuad2Edges(arc4, arc1)
Qface10 = geompy.MakeQuad2Edges(arc5, arc2)
Qface11 = geompy.MakeQuad2Edges(arc6, arc3)
Qface12 = geompy.MakeQuad2Edges(arc10, arc7)
Qface13 = geompy.MakeQuad2Edges(arc11, arc8)
Qface14 = geompy.MakeQuad2Edges(arc12, arc9)
Qface15 = geompy.MakeQuad2Edges(line30, line32)
Qface16 = geompy.MakeQuad2Edges(line35, line33)

#Add Quanrangle faces to study
geompy.addToStudy(Qface1, "Qface1")
geompy.addToStudy(Qface2, "Qface2")
geompy.addToStudy(Qface3, "Qface3")
geompy.addToStudy(Qface4, "Qface4")
geompy.addToStudy(Qface5, "Qface5")
geompy.addToStudy(Qface6, "Qface6")
geompy.addToStudy(Qface7, "Qface7")
geompy.addToStudy(Qface8, "Qface8")
geompy.addToStudy(Qface9, "Qface9")
geompy.addToStudy(Qface10, "Qface10")
geompy.addToStudy(Qface11, "Qface11")
geompy.addToStudy(Qface12, "Qface12")
geompy.addToStudy(Qface13, "Qface13")
```

```
geompy.addToStudy(Qface14, "Qface14")
geompy.addToStudy(Qface15, "Qface15")
geompy.addToStudy(Qface16, "Qface16")

#Make one set of faces
Compound1 = geompy.MakeCompound([Qface1,Qface2,Qface3,
Qface4,Qface5,Qface6,Qface7,Qface8,Qface9,Qface10,
Qface11,Qface12,Qface13,Qface14,Qface15,Qface16])
geompy.addToStudy(Compound1, "Compound1")

#Extrude the set of faces into a 3d body
pr1 = geompy.MakePrismVecH( Compound1,OZ,H)
geompy.addToStudy(pr1, "pr1")

#Update the browser window
salome.sg.updateObjBrowser(True)
```

# APPENDIX B. THE SWAK4FOAM LINES THAT ARE DEFINED IN THE CONTROLDICT.

```
T_massAverage
  {
    type swakExpression;
    valueType      surface;
    surfaceName fin_outletT;
    outputControlMode    timeStep;
    outputInterval  10;
    valueOutput      true;
    enabled true;
    verbose true;
        surface
    {
        type             cuttingPlane;// always triangulated
        planeType pointAndNormal;
            pointAndNormalDict
            {
        basePoint (0.143 0.0066 0.001 );
        normalVector (1 0 0);
            }
    }
        expression
    "(sum(mag(U)*rho*area()*T))/sum(rho*area()*mag(U))" ;
        accumulations (
   average
                        );
}
```

# APPENDIX C. EINSTEIN NOTATION.

The governing equations are expressed with Einsteins summations convention for simple and clear formulation. This formulation is constructed with three main rules:

- Each index can only appear twice in one term
- Repeated indices are implicitly summed over
- Each term must contain a unique index that appears only once in each term

For example hydrostatic pressure, an isotropic pressure that affects only on the three coordinate axis and has no deviatoric(shear) components, can be expressed as follows:

$$-p_{kk} = -\Sigma_{k=1}^{3} p_{kk} = -(p_{11} + p_{22} + p_{33}) = -\begin{Bmatrix} p_{11} & 0 & 0 \\ 0 & p_{22} & 0 \\ 0 & 0 & p_{33} \end{Bmatrix} \tag{1}$$

# APPENDIX D. TURBULENT QUANTITIES.

One of these turbulence quantities is the turbulent kinetic energy $k$ 2. It physically represents the energy of the turbulence per unit mass.

$$k = \frac{1}{2}(\overline{u_i'} \cdot \overline{u_j'}) = \frac{1}{2}(\overline{u'^2} + \overline{v'^2} + \overline{w'^2}) \tag{2}$$

These turbulent velocity components are the ones explained in subsection 5.3.1. Then another important term is the Turbulent intensity that is the ratio of the root mean square of the turbulent fluctuations to the mean flow velocity:

$$I \equiv \frac{u'}{U} = \frac{\sqrt{\frac{1}{3}(u'^2 + v'^2 + w'^2)}}{\sqrt{U_x^2 + U_y^2 + U_z^2}} \tag{3}$$

where $U_x, U_y$ and $U_z$ refers to the three mean velocity components.

Another important turbulent characteristic that is the dissipation of the turbulent energy $\epsilon$ as explained in 5. The exact transport equation for turbulent dissipation can be describe as follows:

$$\epsilon = \nu \overline{\frac{\partial v_i'}{\partial x_j} \frac{\partial v_i'}{\partial x_j}} \tag{4}$$

where $\nu$ is the kinematic viscosity and $v_i'$ represents the turbulent fluctuation. Modeling of the turbulent kinetic energy dissipation is usually done by assuming that the small scale fluctuations are isotropic, meaning the value is independent of the direction. When $\epsilon$ is modelled for example in the basic $k - \epsilon$ model, a boussinesq assumption for turbulent eddy viscosity is being introduced. The turbulent viscosity is then computed as:

$$\nu_t = C_\mu \frac{k^2}{\epsilon} \tag{5}$$

where $C_\mu$ is a constant and k and $\epsilon$ are the just introduced turbulent classifications. This way the turbulent fluctuations are modelled as an introduced increased viscosity. The last turbulent quantity classification we will introduce here is the turbulent kinetic energy specific dissipation which is defined as:

$$\omega = \frac{\epsilon}{C_\mu k} \tag{6}$$

This will be used in a $k - \omega$ model for better prediction of shear stress near the wall. For a more elaborate introduction to different turbulence terms and equations can be found from literature for example from a book by Versteeg and Malalasekera (2007).

# APPENDIX E. TURBULENCE MODELS IN OPENFOAM

| Turbulence models for compressible and incompressible flow available in OpenFOAM | | |
|---|---|---|
| Model call name in openFOAM | Short description | Key characteristics |
| laminar | Dummy turbulence model for laminar flow | Only for low reynolds numbers. |
| kEpsilon | Standard high-$Re$ $k-\epsilon$ model | Two equation model for fully turbulent flow. |
| kOmega | Standard high,$Re$ $k-\omega$ model | Two equation model with wall handling and $k-\epsilon$ in the free stream. |
| kOmegaSST | $k-\omega$-SST(Shear Stress Transport) model | Blending function for wall handling. Transport equation for principle turbulent shear stress. |
| RNGkEpsilon | Renormalisation Group $Re$ $k-\epsilon$ | Renormalisation of the N-S equations to account the effects of smaller scales (Yakhot et al., 1992). Established model for indoor air simulations |
| LienCubicKELowRe | Low Re $k-\epsilon$ | Cubic non-linear low Re |
| LRR | RSM | Reynolds stress model |
| LaunderGibsonRSTMKE | RSM | Reynold stress model + wall reflection |
| realizableKE | Realizable $k-\epsilon$ | More accurate in predicting separation than basic $k-\epsilon$ |
| SpalartAllmara | Spalart-Allmaras mixing length model | Should only be used for external flows |

***Table 1*** *Different turbulence models available for compressible and incompressible flow available in OpenFOAM*

| Turbulence models for compressible *or* incompressible flow available in OpenFOAM | | | |
|---|---|---|---|
| Model call name in openFOAM | $I/C$ | Short description | Key characteristics |
| NonlinearKEShih | I | Non-linear Shih low Reynolds number $k-\epsilon$ | Quadratic non-linear $k-\epsilon$ model. Fundamentally mathematical tensor approach. |
| LienCubicKE | I | Non-linear $k-\epsilon$ | Cubic non-linear $k-\epsilon$ model |
| qZeta | I | Q-z | Gibson Dafa' Alla's Q-zeta model for low Reynolds numbers |
| LaunderSharmaKE | C | Low Re $k-\epsilon$ model | Added damping functions for better $k$ and $\epsilon$ calculation in the viscous sublayer (Launder and Sharma, 1974) Widely used for combustion flows |
| LamBremhorstKE | I | Low Re $k-\epsilon$ | |

***Table 2*** *Different turbulence models for compressible or incompressible flow available in OpenFOAM*